

[illegible]

```

IIIIII  NN    NN  IIIIII  AAAAAA  DDDDDDDD  PPPPPPPP  UU    UU  VV    VV    11
IIIIII  NN    NN  IIIIII  AAAAAA  DDDDDDDD  PPPPPPPP  UU    UU  VV    VV    11
  II    NN    NN  II    AA    AA  DD    DD  PP    PP  UU    UU  VV    VV    11
  II    NN    NN  II    AA    AA  DD    DD  PP    PP  UU    UU  VV    VV    11
  II    NNNN  NN  II    AA    AA  DD    DD  PP    PP  UU    UU  VV    VV    11
  II    NNNN  NN  II    AA    AA  DD    DD  PP    PP  UU    UU  VV    VV    11
  II    NN  NN  NN  II    AA    AA  DD    DD  PPPPPPPP  UU    UU  VV    VV    11
  II    NN  NN  NN  II    AA    AA  DD    DD  PPPPPPPP  UU    UU  VV    VV    11
  II    NN    NN  NN  II    AAAAAAAAAA  DD    DD  PP    PP  UU    UU  VV    VV    11
  II    NN    NN  NN  II    AAAAAAAAAA  DD    DD  PP    PP  UU    UU  VV    VV    11
  II    NN    NN  NN  II    AA    AA  DD    DD  PP    PP  UU    UU  VV    VV    11
  II    NN    NN  NN  II    AA    AA  DD    DD  PP    PP  UU    UU  VV    VV    11
  II    NN    NN  NN  IIIIII  AA    AA  DDDDDDDD  PP    PP  UUUUUUUUUU  VV    VV    11
IIIIII  NN    NN  IIIIII  AA    AA  DDDDDDDD  PP    PP  UUUUUUUUUU  VV    VV    11

```

```

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SSSSSS
LL      II     SSSSSS
LL      II     SS
LL      II     SS
LL      II     SS
LL      II     SS
LLLLLLLL  IIIIII  SSSSSSSS
LLLLLLLL  IIIIII  SSSSSSSS

```

(3)	254	Macros to describe nexus configurations
(4)	379	Adapter-specific data structures
(5)	523	CPU-specific data structures
(6)	723	Message strings
(7)	734	INISMAP, Initialize and map nexuses
(8)	899	INITADP 780, 750, 730, and UV1
(9)	916	CONFIG_IOSPACE
(10)	1066	CREATE_ARRAYS
(11)	1109	MAP_PAGES
(13)	1269	INISUBSPACE
(14)	1339	INISUBADP - BUILD ADP AND INITIALIZE UBA
(14)	1815	INISMBADP - BUILD ADP AND INITIALIZE MBA
(14)	1816	INISDRADP - BUILD ADP AND INITIALIZE DR32
(14)	1817	INISCIADP - BUILD ADP AND INITIALIZE CI
(14)	1997	INISKDZ11
(14)	2031	INISCONSOLE, init data structures for console
(15)	2141	EXESINI_TIMWAIT - COMPUTE CORRECT TIMEWAIT LOOP VALUES
(16)	2299	EXESINIT_TODR - SET SYSTEM TIME TO CORRECT VALUE AT STARTUP


```

0000 1      .NLIST  CND
0000 5
0000 9
0000 13
0000 17
0000 19      .TITLE  INIADPUV1 - ADAPTER INITIALIZATION FOR MICRO-VAX I
0000 21
0000 25
0000 26      .IDENT  'V04-002'
0000 27
0000 28 *****
0000 29 *
0000 30 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 31 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 32 *  ALL RIGHTS RESERVED.
0000 33 *
0000 34 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 35 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 36 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 37 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 38 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 39 *  TRANSFERRED.
0000 40 *
0000 41 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 42 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 43 *  CORPORATION.
0000 44 *
0000 45 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 46 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 47 *
0000 48 *
0000 49 *****
0000 50
0000 51 Facility: System bootstrapping and initialization
0000 52
0000 53 Abstract: This module contains initialization routines that are loaded
0000 54           during system initialization (rather than linked into the system).
0000 55
0000 56 Environment: Mode = KERNEL, Executing on INTERRUPT stack, IPL=31
0000 57
0000 58 Author:  Trudy C. Matthews           Creation date: 22-Jan-1981
0000 59
0000 60 Modification history:
0000 61
0000 62 V04-002 TCM0013           Trudy C. Matthews           10-Sep-1984
0000 63           Add $BQODEF missing from TCM0012.
0000 64
0000 65 V04-001 TCM0012           Trudy C. Matthews           07-Sep-1984
0000 66           For venus processor: turn on cache before calibrating
0000 67           TIMEDWAIT cells (routine EX$INI_TIMWAIT). Store the TIMEDWAIT
0000 68           values calculated after cache is enabled in the boot driver's
0000 69           TIMEDWAIT cells. This is because the boot driver initially
0000 70           has to run with cache off, but after booting will run with
0000 71           cache on.
0000 72
0000 73 V03-024 TCM0011           Trudy C. Matthews           31-Jul-1984
0000 74           Change venus's CRD interrupt vector back to ^X54 in the SCB.

```

```

0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :
0000 80 :
0000 81 :
0000 82 :
0000 83 :
0000 84 :
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :
0000 115 :
0000 116 :
0000 117 :
0000 118 :
0000 119 :
0000 120 :
0000 121 :
0000 122 :
0000 123 :
0000 124 :
0000 125 :
0000 126 :
0000 127 :
0000 128 :
0000 129 :
0000 130 :
0000 131 :

```

```

and its SBIA Fail vector to ^X64.

V03-023 WMC0001 Wayne Cardoza 30-Jul-1984
Add H memory to 780 list.

V03-022 TCM0010 Trudy C. Matthews 25-Jul-1984
Fix a bug in INISUBSPACE for the 11/790 that caused second
and subsequent unibus adapter spaces to be mapped incorrectly.
Fix bugs in INISSCB for the 11/790. Fix conditional
assembly flags in INISCONSOLE for the 11/790.

V03-021 KDM0100 Kathleen D. Morse 01-May-1984
Correct address of memory CSRs to be past the 8 missing
Qbus adapter pages that do not exist.

V03-020 KDM0099 Kathleen D. Morse 27-Apr-1984
On a MicroVAX I, if the sysgen parameter TIMEDWAIT is set
to request no time-prompting, then use the last recorded
system time instead. This is found in EXESGQ_TODCBASE
which can be updated with a SET TIME command.

V03-019 RLRSCORPIO Robert L. Rappaport 16-Mar-1984
Begin additions (to INISIOMAP) for Scorpio support.
Also move ADAPDESC to SYSMAR.MAR, changing it to remove
the ADAP_GENERAL array.

V03-018 RLRINIADP Robert Rappaport 28-Feb-1984
Add refinements to previous update that introduces
longword array CONFREG. Mainly add logic to allow for
independently assembled invocations of ADAPDESC macro
to be linked into this code. This provides possible
support of BI as a public bus, with user defined nodes.

V03-017 KPL0100 Peter Lieberwirth 30-Jan-1984
Implement first step towards a longword-array CONFREG to
replace current byte array CONFREG. INIADP will construct
two confregs, CONFREG and CONFREGL. CONFREGL will be
a longword array. The high byte will be a VMS-bus
designation, and the low word will contain the 16-bit
device type. The BI introduces 16 bit device types.

When all references to CONFREG have been modified to touch
CONFREGL, INIADP will be modified again to stop creating
the byte array.

While here, map 9 pages of CI register space, up from 8.

V03-016 KPL0001 Peter Lieberwirth 17-Jan-1984
Fix bug in V03-015 that caused a failure to boot on 750s.
Specifically, add NDT$_MEM1664NI to ADAPDESC macro.

V03-015 TCM0009 Trudy C. Matthews 12-Dec-1983
Add support for booting from VENUS console device to
INISCONSOLE. When mapping I/O space on VENUS, use the
PAMM to determine if any adaptors are present on the
ABUS.

```


0000	132	:	V03-014	KDM0081	Kathleen D. Morse	13-Sep-1983
0000	133	:		Create version for Micro-VAX I.		
0000	134	:				
0000	135	:	V03-013	DWT0126	David W. Thiel	30-Aug-1983
0000	136	:		Modify EXESINIT_TODR to set internal time without		
0000	137	:		modifying the contents of the system disk.		
0000	138	:				
0000	139	:	V03-012	KDM0062	Kathleen D. Morse	18-Jul-1983
0000	140	:		Add loadable, cpu-dependent routine for initializing		
0000	141	:		the time-wait loop data cells, EXESINI_TIMWAIT.		
0000	142	:				
0000	143	:	V03-011	KDM0057	Kathleen D. Morse	15-Jul-1983
0000	144	:		Added loadable, cpu-dependent routine for initializing		
0000	145	:		the system time, EXESINIT_TODR.		
0000	146	:				
0000	147	:	V03-010	KTA3071	Kerbey T. Altmann	12-Jul-1983
0000	148	:		Include CPU-specific console init code.		
0000	149	:				
0000	150	:	V03-009	TCM0008	Trudy C. Matthews	10-Jan-1983
0000	151	:		Change PSECT of 11/790 data that must stick around after		
0000	152	:		INIADP is deleted. Build arrays ABUS VA, ABUS_TYPE, and		
0000	153	:		ABUS_INDEX that describe the 11/790 ABUS configuration.		
0000	154	:				
0000	155	:	V03-008	MSH0002	Maryann Hinden	08-Dec-1982
0000	156	:		Add powerfail support for DW750.		
0000	157	:				
0000	158	:	V03-007	ROW0142	Ralph O. Weber	24-NOV-1982
0000	159	:		Change UBA interrupt services routines prototype so that		
0000	160	:		UBAERRADR is correctly computed as an offset from UBAINTRASE.		
0000	161	:				
0000	162	:	V03-006	TCM0007	Trudy C. Matthews	10-Nov-1982
0000	163	:		Add 11/790-specific initialization of SCB.		
0000	164	:				
0000	165	:	V03-005	TCM0006	Trudy C. Matthews	8-Nov-1982
0000	166	:		Initialize field ADP\$L_AVECTOR with the address of		
0000	167	:		each adapter's first SCB vector.		
0000	168	:				
0000	169	:	V03-004	KTA3018	Kerbey T. Altmann	30-Oct-1982
0000	170	:		Move from INILOA facility, rename from INITADP,		
0000	171	:		put in conditional assembly, rewrite some routines.		
0000	172	:				
0000	173	:	V03-003	MSH0001	Maryann Hinden	24-Sep-1982
0000	174	:		Change EXESDW780_INT to EXESUBAERR_INT.		
0000	175	:				
0000	176	:	V03-002	TCM0005	Trudy C. Matthews	10-Aug-1982
0000	177	:		Added support for 11/790 processor.		
0000	178	:				
0000	179	:	V03-001	KDM0002	Kathleen D. Morse	28-Jun-1982
0000	180	:		Added \$DCDEF.		
0000	181	:				
0000	182	---				

0000	184	:		
0000	185	:	MACRO LIBRARY CALLS	
0000	186	:		
0000	187	:	\$ADPDEF	: Define ADP offsets.
0000	188	:	\$BIICDEF	: Define BIIC offsets.
0000	189	:	\$BQODEF	: Define boot vector offsets.
0000	190	:	\$BTODEF	: Define boot devices
0000	191	:	\$BUADEF	: Define BUA Register offsets.
0000	192	:	\$CRBDEF	: Define CRB offsets.
0000	193	:	\$DCDEF	: Define adapter types
0000	194	:	\$DDBDEF	: Define DDB offsets
0000	195	:	\$DYNDDEF	: Define data structure type codes.
0000	196	:	\$IDBDEF	: Define interrupt dispatcher offsets.
0000	213	:	\$IOUV1DEF	: Define Micro-VAX I I/O space.
0000	219	:	\$MCHKDEF	: Define machine check masks.
0000	220	:	\$NDTDEF	: Define nexus device types.
0000	221	:	\$PRDEF	: Define IPR numbers.
0000	222	:		
0000	226	:		
0000	230	:		
0000	234	:		
0000	238	:		
0000	240	:	\$PRUV1DEF	: Define Micro-VAX I specific IPRs.
0000	242	:		
0000	246	:		
0000	247	:	\$PTEDEF	: Define Page Table Entry bits.
0000	248	:	\$RPBDEF	: Define Restart Parameter Block fields.
0000	249	:	\$UBADEF	: Define UBA register offsets.
0000	250	:	\$UCBDEF	: Define UCB offsets.
0000	251	:	\$VADEF	: Define virtual address fields.
0000	252	:	\$VECDEF	: Define vec offsets.


```
0000 254 .SBTTL Macros to describe nexus configurations
0000 255
0000 256 The macros FLOAT_NEXUS and FIXED_NEXUS add one or more entries to a
0000 257 nexus descriptor table. Each entry is of the form:
0000 258
0000 259 +-----+
0000 260 | PFN of nexus I/O space |
0000 261 +-----+
0000 262 | bus | 0 | type |
0000 263 +-----+
0000 264 type = 0 -> floating nexus
0000 265 type = non-zero -> fixed nexus; type = fixed adapter type
0000 266 bus = 0, if SBI; %x80 if BI (this is a VMS-only designation)
0000 267
0000 268 device_type: SBI adapters have 8-bit device type codes. These
0000 269 device types are simple integers.
0000 270
0000 271 BI adapters have 16-bit device type codes, that are
0000 272 subject to the following interpretation:
0000 273
0000 274 - the MSB of the device-type field will be 0 for DEC
0000 275 devices and 1 for non-DEC devices,
0000 276
0000 277 - DEC memory devices will have 0s in the high-order
0000 278 byte of the device type,
0000 279
0000 280 - non-DEC supplied memory devices will have a 1 in the
0000 281 MSB of the high-order byte, and the rest of the high
0000 282 order byte will contain 0s.
0000 283
0000 284 - The "all 0s" and "all 1s" device-type codes are
0000 285 reserved for DEC.
0000 286
0000 287 If SBI type codes were simply expanded to a word for purposes of the routines
0000 288 in this module, there would be possible conflicts between SBI devices and
0000 289 BI memory adapters supplied by DEC. Voila: the bus type.
0000 290
0000 291 Macro FLOAT_NEXUS.
0000 292 INPUTS:
0000 293 PHYSADR -- physical address of 1 or more contiguous floating nexus
0000 294 slots
0000 295 NUMNEX -- number of contiguous floating nexuses, default = 1
0000 296 PERNEX -- amount of address space per nexus (does not have to be
0000 297 specified if NUMNEX = 1)
0000 298
0000 299 .MACRO FLOAT_NEXUS PHYSADR,NUMNEX=1,PERNEX=0
0000 300 PA = PHYSADR
0000 301 .REPEAT NUMNEX ; For each nexus...
0000 302 .LONG <PA/^X200> ; Store PFN.
0000 303 .LONG 0 ; Store floating nexus type.
0000 304 PA = PA + PERNEX ; Increment to physical address of next nexus.
0000 305 .ENDR
0000 306 .ENDM FLOAT_NEXUS
0000 307
0000 308 ; Macro FIXED_NEXUS.
0000 309 ;
0000 310 ;
```



```

0000 311 : INPUTS:
0000 312 : PHYSADR - physical address of 1 or more contiguous fixed nexus slots
0000 313 : PERNEX - amount of address space per nexus
0000 314 : NEXUSTYPES - a list of fixed nexus types, enclosed in <>
0000 315 :
0000 316 : .MACRO FIXED_NEXUS PHYSADR,PERNEX=0,NEXUSTYPES
0000 317 : PA = PHYSADR
0000 318 : .IRP TYPECODE,NEXUSTYPES ; For each fixed nexus type...
0000 319 : .LONG <PA/^X200> ; Store PFN.
0000 320 : .LONG TYPECODE ; Store fixed nexus type.
0000 321 : PA = PA + PERNEX ; Increment to address of next nexus.
0000 322 : .ENDR
0000 323 : .ENDM FIXED_NEXUS
0000 324 :
0000 325 :
0000 326 : Macro NEXUSDESC_TABLE - declare the beginning of a NEXUS descriptor table
0000 327 :
0000 328 : 1st byte in table (at offset -5 from label) contains length of
0000 329 : adapter type code field in CSR's on this bus. [Note for SBI like
0000 330 : busses, this is 1.] The next longword (at offset -4) in the
0000 331 : table contains the Software defined bus type byte defined in the
0000 332 : high order byte of the longword. [Note for SBI like busses, this
0000 333 : value is 0, for the BI it is ^x80.]
0000 334 :
0000 335 :
0000 336 : Define parameters that may be specified or used in macro invocation.
0000 337 :
00000000 0000 338 BI_LIKE = 0 ; BI like bus.
00000001 0000 339 SBI_LIKE = 1 ; SBI like bus.
0000 340 :
00000001 0000 341 SBI_CSR_LEN = 1 ; Length of type code field in adapter CSR's
0000 342 : on SBI, CMI, etc.
00000002 0000 343 BI_CSR_LEN = 2 ; Length of type code field in adapter CSR's
0000 344 : on BI.
0000 345 :
00000000 0000 346 SBI_BUS_CODE = 0 ; Software defined bus code for SBI like busses.
80000000 0000 347 BI_BUS_CODE = ^x80000000 ; Software defined bus code for the BI.
0000 348 :
0000 349 : .MACRO NEXUSDESC_TABLE LABEL,BUS_TYPE=SBI_LIKE
0000 350 : .IF EQ,BUS_TYPE-SBI_LIKE
0000 351 : .BYTE SBI_CSR_LEN
0000 352 : .LONG SBI_BUS_CODE
0000 353 : .IFF
0000 354 : .IF EQ,BUS_TYPE-BI_LIKE
0000 355 : .BYTE BI_CSR_LEN
0000 356 : .LONG BI_BUS_CODE
0000 357 : .IFF
0000 358 : .ERROR ; UNRECOGNIZED BUS TYPE, NEXUSDESC_TABLE;
0000 359 : .ENDC
0000 360 : .ENDC
0000 361 :
0000 362 LABEL:
0000 363 : .ENDM NEXUSDESC_TABLE
0000 364 :
FFFFF0FB 0000 365 CSR_LEN_OFFSET = -5 ; Offset before nexus descriptor of
0000 366 : byte containing length of adapter
0000 367 : type field in adapter CSR.

```

```

FFFFFFFFC 0000 368 BUS_CODE_OFFSET = -4          ; Offset before nexus descriptor table
           0000 369                               ; of longword containing software
           0000 370                               ; defined bus type to be or'ed with
           0000 371                               ; adapter type to produce NDT$_ value.
           0000 372 :
           0000 373 : Macro END_NEXUSDESC.
           0000 374 :
           0000 375 .MACRO END_NEXUSDESC
           0000 376 .LONG 0
           0000 377 .ENDM END_NEXUSDESC          ; PFN=0 -> end of nexus descriptors.

```

```

0000 379      .SBTTL Adapter-specific data structures
0000 380      ;
0000 381      ; Put a symbol for arrays built by macros in the correct psects.
0000 382      ;
0000 383      ;***** ADAPTERS array *****
00000000 384      .PSECT $$$INIT$DATA0
0000 385 ADAPTERS:                                ; Build adapter type code arrays here.
0000 386
00000000 387      .PSECT $$$INIT$DATA1                ; User contributions in this .PSECT.
0000 388      ; End of ADAPTERS array.
0000 389      ;***** End of ADAPTERS array *****
0000 390
0000 391      ;***** NUM_PAGES array *****
00000000 392      .PSECT $$$INIT$DATA2
0000 393 NUM_PAGES:                                ; Build 'number of pages to map' array.
00000000 394      .PSECT $$$INIT$DATA3                ; User contributions in this .PSECT.
0000 395      ;***** End of NUM_PAGES array *****
0000 396
0000 397      ;***** INIT_ROUTINES array *****
00000000 398      .PSECT $$$INIT$DATA4
0000 399 INIT_ROUTINES:                            ; Build 'address of init routine' array.
00000000 400      .PSECT $$$INIT$DATA5                ; User contributions in this .PSECT.
0000 401      ;***** End of INIT_ROUTINES array *****
0000 402
0000 403      ;
0000 404      ; To add a new adapter type:
0000 405      ; 1) Add a new ADAPDESC macro invocation to the end of this list.
0000 406      ;
00000000 407      .PSECT $$$INIT$DATA, LONG
0000 408
0000 409      ;
0000 410      ; Default interrupt vectors for UNIBUS system devices
0000 411      ; (This array is indexed by the RPB field RPB$B_DEVTYPE, if the RPB field
0000 412      ; RPB$W_ROUBVEC is zero. If RPB$W_ROUBVEC is not zero, then RPB$W_ROUBVEC
0000 413      ; is used and this array is not referenced at all. RPB$W_ROUBVEC is set up
0000 414      ; by PQDRIVER. RPB$L_BOOTRO is set by VMB to contain the device name in
0000 415      ; ASCII, not the vector number and device type, as it does on full
0000 416      ; architecture VAX machines.
0000 417      ;
0000 418 BOOTVECTOR:
0088 0000 419      .WORD    ^X88                        ; RK06/7 Interrupt vector
0070 0002 420      .WORD    ^X70                        ; RL01/2 Interrupt vector
0004 421
0004 422 BUS_CSR_LEN:                                ; Static byte containing the length (in bytes)
00 0004 423      .BYTE    0                                ; of the adapter type field in the CSR's of
0005 424      ; the bus currently being configured. The
0005 425      ; proper value for the bus of interest is
0005 426      ; copied here, from the current nexus
0005 427      ; descriptor table, when we enter subroutine
0005 428      ; CONFIG_IOSPACE.
0005 429
0005 430 SW_BUS_CODE:                                ; Static longword containing the software
00000000 0005 431      .LONG    0                                ; defined bus type, of the bus currently being
0009 432      ; configured, in the high order byte. The
0009 433      ; proper value for the bus of current interest
0009 434      ; is copied here, from the nexus descriptor
0009 435      ; table, when we enter subroutine

```



```

0009 436 ; CONFIG_IOSPACE.
0009 437
0009 438 DIRECT_VEC_NODE_CNT: ; Static longword that counts the number of
0009 439 ; direct vectoring adapter nodes that we have
00000000 0009 440 .LONG 0 ; run across so far.
0000 441
00000001 0000 442 $$$VMSDEFINED = 1 ; Define symbol that means VMS system software.
00000080 0000 443 NUMUBAVEC = 128 ; ALLOW FOR 128 UNIBUS VECTORS
0000 444
0000 445 ADAPDESC - ; Memory. ** MUST BE 1ST IN DESCRIPTOR LIST **
0000 446 ADPTYPES=<NDT$_MEM1664NI,NDT$_MEM4NI,NDT$_MEM4I,NDT$_MEM16NI, -
0000 447 NDT$_MEM16I, -
0000 448 NDT$_MEM64NIL,NDT$_MEM64EIL,NDT$_MEM64NIU,NDT$_MEM64EIU, -
0000 449 NDT$_MEM64I, -
0000 450 NDT$_MEM256NIL,NDT$_MEM256EIL,NDT$_MEM256NIU,NDT$_MEM256EIU, -
0000 451 NDT$_MEM256I, -
0000 452 NDT$_SCORMEM> -
0000 453 Numpages=1
0000 454
0000 455 ADAPDESC - ; MASSbus.
0000 456 ADPTYPES=NDT$_MB, -
0000 457 Numpages=8, -
0000 458 INITRTN=INI$MBADP
0000 459
0000 460 ADAPDESC - ; UNibus.
0000 461 ADPTYPES=<NDT$_UB0,NDT$_UB1,NDT$_UB2,NDT$_UB3,NDT$_BUA>, -
0000 462 Numpages=8, -
0000 463 INITRTN=INI$SUBSPACE
0000 464
0000 465 ADAPDESC - ; Multi-port memory.
0000 466 ADPTYPES=<NDT$_MPM0,NDT$_MPM1,NDT$_MPM2,NDT$_MPM3>, -
0000 467 Numpages=1, -
0000 468 INITRTN=INI$MPMADP
0000 469
0000 470 ADAPDESC - ; DR32.
0000 471 ADPTYPES=NDT$_DR32, -
0000 472 Numpages=4, -
0000 473 INITRTN=INI$DRADP
0000 474
0000 475 ADAPDESC - ; C1780
0000 476 ADPTYPES=NDT$_CI, -
0000 477 Numpages=9, -
0000 478 INITRTN=INI$CIADP
0000 479
0000 480 ADAPDESC - ; KDZ11 Processor
0000 481 ADPTYPES=NDT$_KDZ11, -
0000 482 Numpages=1, -
0000 483 INITRTN=INI$KDZ11
0000 484

```

```

0000 523      .SBTTL  CPU-specific data structures
0000 524      :
0000 525      : To add a new CPU type:
0000 526      : 1) Create a new nexus descriptor table, using FLOAT_NEXUS and
0000 527      :    FIXED_NEXUS macros.  Put an END_NEXUSDESC macro at the end.
0000 528      :
0000 529      :
0000 532      :
0000 590      :
0000 617      :
0000 659      :
0000 660      :
0000 662      CPU_ADPSIZE:
0258 0000 663      .WORD    ADPSC_UBAADPLEN
000F 664      :
000F 665      :
000F 666      :
000F 667      : Declare the beginning of a nexus-descriptor table.
000F 668      :
000F 669      NEXUSDESC_TABLE LABEL=NEXUSDESC
0014 670      :
0014 671      :
0014 672      :
0014 673      : Describe all nexuses on a Micro-VAX I processor.
0014 674      :
00000000 0014 675      SBI_CPU = 0
00000000 0014 676      BI_CPU  = 0
0014 677      FIXED_NEXUS =
0014 678      PHYSADR=IOUV1$AL_QBOSP, -
0014 679      NEXUSTYPES=NDT$_DBO
001C 680      END_NEXUSDESC
0020 682      :
0020 706      :
0020 707      :
0020 708      : Nexus "descriptor" arrays -- these arrays hold the nexus-device type and
0020 709      : virtual address of every adapter on the system.  The arrays, CONFREG and
0020 710      : SBICONF, are allocated enough space to hold the maximum number of adapters
0020 711      : that can be attached to any CPU.  When the code discovers how many adapters
0020 712      : actually exist on the system, it will allocate space from non-paged pool
0020 713      : and move a permanent copy of these arrays into that space.
0020 714      :
00000040 0020 715      MAXNEXUS = 64
00000060 0020 716      CONFREG:                                ; Byte array of nexus-device type codes..
00000060 0020 717      .BLKB    MAXNEXUS
00000160 0060 718      SBICONF:                                ; Longword array of VAs of adapter space.
00000160 0060 719      .BLKL    MAXNEXUS
00000260 0160 720      CONFREG:                                ; Longword array of nexus-device type codes
00000260 0160 721      .BLKL    MAXNEXUS

```

INIADPUV1
V04-002

H 15
- ADAPTER INITIALIZATION FOR MICRO-VAX I 16-SEP-1984 01:04:35 VAX/VMS Macro V04-00
Message strings 11-SEP-1984 16:29:18 [SYSLOA.SRC]INIADP.MAR;3

Page 11
(6)

```
0000000D 0260 723 .SBTTL Message strings
0000000A 0260 724
0260 725 CR = 13
0260 726 LF = 10
0260 727 NOSPT:
0260 728 .ASCIIZ <CR><LF>/%EXECINIT-F-Insufficient SPT entries/<CR><LF>
026C
0278
0284
```

2D	54	49	4E	49	43	45	58	45	25	0A	0D
65	69	63	69	66	66	75	73	6E	49	2D	46
69	72	74	6E	65	20	54	50	53	20	74	6E
							00	0A	0D	73	65


```

0289 734 .SBTTL INISMAP, Initialize and map nexuses
0289 735
0289 736 ++
0289 737 FUNCTIONAL DESCRIPTION:
0289 738 This routine is executed only once, during system initialization.
0289 739 It loops through all nexuses on the system, testing for
0289 740 adapters. When it finds an adapter, it maps its I/O space and
0289 741 initializes it.
0289 742
0289 743 INPUTS:
0289 744 BOOSGL_SPTFREL - next free VPN
0289 745 MMG$GL_SPTVASE - base of system page table
0289 746 EXE$GL_RPB - address of reboot parameter block
0289 747
0289 750 OUTPUTS:
0289 751 R0 - $$$_NORMAL
0289 752
0289 753 For each adapter found, its accessible I/O space is mapped to virtual
0289 754 addresses. An ADP (Adapter Control Block) is built, and the hardware
0289 755 adapter is initialized.
0289 756
0289 757 The arrays CONFREG (a byte array of nexus-device type codes, defined
0289 758 by NDT$ symbols) and SBICONF (a longword array of
0289 759 virtual addresses that map adapter space) are initialized. Pointers
0289 760 to these arrays are stored in EXE$GL_CONFREG and
0289 761 MMG$GL_SBICONF. The number of entries in these two parallel arrays is
0289 762 stored in EXE$GL_NUMNEXUS.
0289 763
0289 764 Since BI devices have a 16-bit device type code, a new CONFREG array is
0289 765 constructed. This is a longword array called CONFREG_L.
0289 766
0289 767 Several locations in the RPB that describe the boot device are init'ed:
0289 768 RPB$L_BOOTR1 - holds index into CONFREG and SBICONF for the boot
0289 769 adapter
0289 770 RPB$L_ADPVIR - holds VA of boot device adapter's register space
0289 771 RPB$L_CSRVIR - holds VA of boot device's register space
0289 772
0289 773
00000000 774 .PSECT $$$INIT$CODE,QUAD
0000 775 INISMAP::
0000 776
00FF 8F BB 0000 777 PUSHR #*M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
0004 778
0004 779 Set up common inputs to CONFIG_IOSPACE subroutine for the CPU-specific code.
0004 780
0004 781 MOVL G*BOOSGL_SPTFREL,R2 : Get next available VPN.
0004 782 MOVL G*MMG$GL_SPTBASE,R3 : Get base of System Page Table.
0004 783 MOVAL (R3)[R2],R3 : Compute SVASPT.
0004 784 ASHL #9,R2,R2 : Convert VPN to VA.
0004 785 BISL #VASM_SYSTEM,R2 : Set system bit.
0004 786 CLRL R4 : Clear index into CONFREG and SBICONF.
0004 787 MOVL G*EXE$GL_RPB,R9 : Get address of RPB.
0004 791 MOVAL W*SBICONF,G*MMG$GL_SBICONF : Set pointers to local copies
0004 792 MOVAL W*CONFREG,G*EXE$GL_CONFREG : of these arrays for init routines.
0004 793 MOVAL W*CONFREG_L,G*EXE$GL_CONFREG_L : ...

```

```

      0045 899      .SBTTL INITADP_780, _750, _730, and _UV1
      0045 900      :
      0045 901      : I/O address space for the 11/780, 11/750, 11/730, and Micro-VAX I cpus
      0045 902      : is statically defined in their respective nexus descriptor tables.
      0045 903      :
      56 0014'CF DE 0045 904      MOVAL W^NEXUSDESC,R6      : Get address of nexus table.
           SB D4 004A 905      CLRL R11      : Signal use 1st page of SCB.
           OB 10 004C 906      BSBB CONFIG_IOSPACE      : Configure processor I/O space.
                004E 907
                004E 909
                0079 30 004E 910      BSBW CREATE_ARRAYS      : Create CONFREG and SBICONF arrays.
      OFFF 8F BA 0051 911      POPR #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11>
      50 01 D0 0055 912      MOVL #1,R0      : Set success status
                05 0058 913      RSB      : Return.

```

```

0059 916      .SBTTL CONFIG_IOSPACE
0059 917      :
0059 918      CONFIG_IOSPACE
0059 919      : Given a nexus descriptor table, which describes what 'nexuses' or
0059 920      : 'slots' are available on a system to hold I/O adapters, find and
0059 921      : initialize all adapters on the system.
0059 922      :
0059 923      Inputs:
0059 924      : R2 - next available virtual address, to be used for mapping I/O space
0059 925      : R3 - address of PTE associated with VA in R2
0059 926      : R4 - Current index into CONFREG and SBICONF arrays (should be 0 the
0059 927      : first time CONFIG_IOSPACE is called)
0059 928      : R6 - address of nexus descriptor table
0059 929      : R9 - address of Restart Parameter Block (RPB)
0059 930      : R10 - PFN of boot adapter space
0059 931      : R11 - page offset from beginning of SCB; tells which page of the SCB
0059 932      : to use for this set of nexuses (passed to routines that init ADP)
0059 933      :
0059 934      Outputs:
0059 935      : R2,R3,R4 - updated
0059 936      : R9,R10,R11 - preserved; all other registers potentially modified
0059 937      : CONFREG - initialized with adapter NDIS code for each nexus
0059 938      : SBICONF - initialized with adapter space VA for each nexus
0059 939      :
0059 940      CONFIG_IOSPACE:
0059 946      :
0059 947      : There is only one adapter, the Qbus.
0059 948      :
0059 949      :
0059 950      :
0059 951      MOVB    CSR_LEN_OFFSET(R6),-      ; Move length of adapter type field
0059 952      W^BUS_CSR_LEN                      ; in CSR's to static location.
0059 953      MOVL    BUS_CODE_OFFSET(R6),-      ; Move software defined bus type code
0059 954      W^SW_BUS_CODE                     ; to static longword.
0059 955      :
0059 956      NXT_NEXUS:                          ; For each nexus...
0059 957      MOVL    (R6)+,R8                  ; Get PFN of nexus.
0059 958      :
0059 959      : Execution continues here if adapter was present.
0059 960      :
0059 961      :
0059 962      GET_TYPE:
0059 963      MOVL    (R6)+,R7                      ; Get nexus-device type from nexus table.
0059 964      :
0059 965      : Here R7 has hardware adapter code or'ed with software bus code.
0059 966      : Translate specific nexus device type code into general adapter type code.
0059 967      :
0059 968      :
0059 969      GET_GEN_TYPE:
0059 970      MOVB    R7,W^CONFREG[R4]           ; Save nexus-device type in CONFREG.
0059 971      MOVL    R7,W^CONFREGL[R4]         ; CONFREGL also filled in.
0059 972      CLRL    R5                          ; Clear loop index.
0059 973      :
0059 974      30$:
0059 975      MOVAL    W^ADAPTERS[R5],R0           ; Get address of adapter type code.
0059 976      PUSHAB   W^NUM_PAGES                ; Push addr of end of ADAPTERS array.
0059 977      CMPL    R0,(SP)+                  ; See if we went beyond array.
0059 978      BGEQU    END_NEXUS                 ; unrecognized adapter, do not map.
0059 979      CMPL    R7,(R0)                  ; Adapter type match?
0059 980      BEQL    40$                       ; If EQL yes, adapter type match.
0059 981      INCL    R5                          ; Increment loop index.
0059 982      :
0059 983      :
0059 984      :
0059 985      :
0059 986      :
0059 987      :
0059 988      :
0059 989      :
0059 990      :
0059 991      :
0059 992      :
0059 993      :
0059 994      :
0059 995      :
0059 996      :
0059 997      :
0059 998      :
0059 999      :
0059 1000     :
0059 1001     :
0059 1002     :
0059 1003     :
0059 1004     :
0059 1005     :
0059 1006     :
0059 1007     :
0059 1008     :
0059 1009     :
0059 1010     :
0059 1011     :
0059 1012     :
0059 1013     :
0059 1014     :
0059 1015     :
0059 1016     :
0059 1017     :
0059 1018     :
0059 1019     :
0059 1020     :
0059 1021     :
0059 1022     :
0059 1023     :
0059 1024     :
0059 1025     :
0059 1026     :
0059 1027     :
0059 1028     :
0059 1029     :
0059 1030     :
0059 1031     :
0059 1032     :
0059 1033     :
0059 1034     :
0059 1035     :
0059 1036     :
0059 1037     :
0059 1038     :
0059 1039     :
0059 1040     :
0059 1041     :
0059 1042     :
0059 1043     :
0059 1044     :
0059 1045     :
0059 1046     :
0059 1047     :
0059 1048     :
0059 1049     :
0059 1050     :
0059 1051     :
0059 1052     :
0059 1053     :
0059 1054     :
0059 1055     :
0059 1056     :
0059 1057     :
0059 1058     :
0059 1059     :
0059 1060     :
0059 1061     :
0059 1062     :
0059 1063     :
0059 1064     :
0059 1065     :
0059 1066     :
0059 1067     :
0059 1068     :
0059 1069     :
0059 1070     :
0059 1071     :
0059 1072     :
0059 1073     :
0059 1074     :
0059 1075     :
0059 1076     :
0059 1077     :
0059 1078     :
0059 1079     :
0059 1080     :
0059 1081     :
0059 1082     :
0059 1083     :
0059 1084     :
0059 1085     :
0059 1086     :
0059 1087     :
0059 1088     :
0059 1089     :
0059 1090     :
0059 1091     :
0059 1092     :
0059 1093     :
0059 1094     :
0059 1095     :
0059 1096     :
0059 1097     :
0059 1098     :
0059 1099     :
0059 1100     :
0059 1101     :
0059 1102     :
0059 1103     :
0059 1104     :
0059 1105     :
0059 1106     :
0059 1107     :
0059 1108     :
0059 1109     :
0059 1110     :
0059 1111     :
0059 1112     :
0059 1113     :
0059 1114     :
0059 1115     :
0059 1116     :
0059 1117     :
0059 1118     :
0059 1119     :
0059 1120     :
0059 1121     :
0059 1122     :
0059 1123     :
0059 1124     :
0059 1125     :
0059 1126     :
0059 1127     :
0059 1128     :
0059 1129     :
0059 1130     :
0059 1131     :
0059 1132     :
0059 1133     :
0059 1134     :
0059 1135     :
0059 1136     :
0059 1137     :
0059 1138     :
0059 1139     :
0059 1140     :
0059 1141     :
0059 1142     :
0059 1143     :
0059 1144     :
0059 1145     :
0059 1146     :
0059 1147     :
0059 1148     :
0059 1149     :
0059 1150     :
0059 1151     :
0059 1152     :
0059 1153     :
0059 1154     :
0059 1155     :
0059 1156     :
0059 1157     :
0059 1158     :
0059 1159     :
0059 1160     :
0059 1161     :
0059 1162     :
0059 1163     :
0059 1164     :
0059 1165     :
0059 1166     :
0059 1167     :
0059 1168     :
0059 1169     :
0059 1170     :
0059 1171     :
0059 1172     :
0059 1173     :
0059 1174     :
0059 1175     :
0059 1176     :
0059 1177     :
0059 1178     :
0059 1179     :
0059 1180     :
0059 1181     :
0059 1182     :
0059 1183     :
0059 1184     :
0059 1185     :
0059 1186     :
0059 1187     :
0059 1188     :
0059 1189     :
0059 1190     :
0059 1191     :
0059 1192     :
0059 1193     :
0059 1194     :
0059 1195     :
0059 1196     :
0059 1197     :
0059 1198     :
0059 1199     :
0059 1200     :
0059 1201     :
0059 1202     :
0059 1203     :
0059 1204     :
0059 1205     :
0059 1206     :
0059 1207     :
0059 1208     :
0059 1209     :
0059 1210     :
0059 1211     :
0059 1212     :
0059 1213     :
0059 1214     :
0059 1215     :
0059 1216     :
0059 1217     :
0059 1218     :
0059 1219     :
0059 1220     :
0059 1221     :
0059 1222     :
0059 1223     :
0059 1224     :
0059 1225     :
0059 1226     :
0059 1227     :
0059 1228     :
0059 1229     :
0059 1230     :
0059 1231     :
0059 1232     :
0059 1233     :
0059 1234     :
0059 1235     :
0059 1236     :
0059 1237     :
0059 1238     :
0059 1239     :
0059 1240     :
0059 1241     :
0059 1242     :
0059 1243     :
0059 1244     :
0059 1245     :
0059 1246     :
0059 1247     :
0059 1248     :
0059 1249     :
0059 1250     :
0059 1251     :
0059 1252     :
0059 1253     :
0059 1254     :
0059 1255     :
0059 1256     :
0059 1257     :
0059 1258     :
0059 1259     :
0059 1260     :
0059 1261     :
0059 1262     :
0059 1263     :
0059 1264     :
0059 1265     :
0059 1266     :
0059 1267     :
0059 1268     :
0059 1269     :
0059 1270     :
0059 1271     :
0059 1272     :
0059 1273     :
0059 1274     :
0059 1275     :
0059 1276     :
0059 1277     :
0059 1278     :
0059 1279     :
0059 1280     :
0059 1281     :
0059 1282     :
0059 1283     :
0059 1284     :
0059 1285     :
0059 1286     :
0059 1287     :
0059 1288     :
0059 1289     :
0059 1290     :
0059 1291     :
0059 1292     :
0059 1293     :
0059 1294     :
0059 1295     :
0059 1296     :
0059 1297     :
0059 1298     :
0059 1299     :
0059 1300     :
0059 1301     :
0059 1302     :
0059 1303     :
0059 1304     :
0059 1305     :
0059 1306     :
0059 1307     :
0059 1308     :
0059 1309     :
0059 1310     :
0059 1311     :
0059 1312     :
0059 1313     :
0059 1314     :
0059 1315     :
0059 1316     :
0059 1317     :
0059 1318     :
0059 1319     :
0059 1320     :
0059 1321     :
0059 1322     :
0059 1323     :
0059 1324     :
0059 1325     :
0059 1326     :
0059 1327     :
0059 1328     :
0059 1329     :
0059 1330     :
0059 1331     :
0059 1332     :
0059 1333     :
0059 1334     :
0059 1335     :
0059 1336     :
0059 1337     :
0059 1338     :
0059 1339     :
0059 1340     :
0059 1341     :
0059 1342     :
0059 1343     :
0059 1344     :
0059 1345     :
0059 1346     :
0059 1347     :
0059 1348     :
0059 1349     :
0059 1350     :
0059 1351     :
0059 1352     :
0059 1353     :
0059 1354     :
0059 1355     :
0059 1356     :
0059 1357     :
0059 1358     :
0059 1359     :
0059 1360     :
0059 1361     :
0059 1362     :
0059 1363     :
0059 1364     :
0059 1365     :
0059 1366     :
0059 1367     :
0059 1368     :
0059 1369     :
0059 1370     :
0059 1371     :
0059 1372     :
0059 1373     :
0059 1374     :
0059 1375     :
0059 1376     :
0059 1377     :
0059 1378     :
0059 1379     :
0059 1380     :
0059 1381     :
0059 1382     :
0059 1383     :
0059 1384     :
0059 1385     :
0059 1386     :
0059 1387     :
0059 1388     :
0059 1389     :
0059 1390     :
0059 1391     :
0059 1392     :
0059 1393     :
0059 1394     :
0059 1395     :
0059 1396     :
0059 1397     :
0059 1398     :
0059 1399     :
0059 1400     :
0059 1401     :
0059 1402     :
0059 1403     :
0059 1404     :
0059 1405     :
0059 1406     :
0059 1407     :
0059 1408     :
0059 1409     :
0059 1410     :
0059 1411     :
0059 1412     :
0059 1413     :
0059 1414     :
0059 1415     :
0059 1416     :
0059 1417     :
0059 1418     :
0059 1419     :
0059 1420     :
0059 1421     :
0059 1422     :
0059 1423     :
0059 1424     :
0059 1425     :
0059 1426     :
0059 1427     :
0059 1428     :
0059 1429     :
0059 1430     :
0059 1431     :
0059 1432     :
0059 1433     :
0059 1434     :
0059 1435     :
0059 1436     :
0059 1437     :
0059 1438     :
0059 1439     :
0059 1440     :
0059 1441     :
0059 1442     :
0059 1443     :
0059 1444     :
0059 1445     :
0059 1446     :
0059 1447     :
0059 1448     :
0059 1449     :
0059 1450     :
0059 1451     :
0059 1452     :
0059 1453     :
0059 1454     :
0059 1455     :
0059 1456     :
0059 1457     :
0059 1458     :
0059 1459     :
0059 1460     :
0059 1461     :
0059 1462     :
0059 1463     :
0059 1464     :
0059 1465     :
0059 1466     :
0059 1467     :
0059 1468     :
0059 1469     :
0059 1470     :
0059 1471     :
0059 1472     :
0059 1473     :
0059 1474     :
0059 1475     :
0059 1476     :
0059 1477     :
0059 1478     :
0059 1479     :
0059 1480     :
0059 1481     :
0059 1482     :
0059 1483     :
0059 1484     :
0059 1485     :
0059 1486     :
0059 1487     :
0059 1488     :
0059 1489     :
0059 1490     :
0059 1491     :
0059 1492     :
0059 1493     :
0059 1494     :
0059 1495     :
0059 1496     :
0059 1497     :
0059 1498     :
0059 1499     :
0059 1500     :
0059 1501     :
0059 1502     :
0059 1503     :
0059 1504     :
0059 1505     :
0059 1506     :
0059 1507     :
0059 1508     :
0059 1509     :
0059 1510     :
0059 1511     :
0059 1512     :
0059 1513     :
0059 1514     :
0059 1515     :
0059 1516     :
0059 1517     :
0059 1518     :
0059 1519     :
0059 1520     :
0059 1521     :
0059 1522     :
0059 1523     :
0059 1524     :
0059 1525     :
0059 1526     :
0059 1527     :
0059 1528     :
0059 1529     :
0059 1530     :
0059 1531     :
0059 1532     :
0059 1533     :
0059 1534     :
0059 1535     :
0059 1536     :
0059 1537     :
0059 1538     :
0059 1539     :
0059 1540     :
0059 1541     :
0059 1542     :
0059 1543     :
0059 1544     :
0059 1545     :
0059 1546     :
0059 1547     :
0059 1548     :
0059 1549     :
0059 1550     :
0059 1551     :
0059 1552     :
0059 1553     :
0059 1554     :
0059 1555     :
0059 1556     :
0059 1557     :
0059 1558     :
0059 1559     :
0059 1560     :
0059 1561     :
0059 1562     :
0059 1563     :
0059 1564     :
0059 1565     :
0059 1566     :
0059 1567     :
0059 1568     :
0059 1569     :
0059 1570     :
0059 1571     :
0059 1572     :
0059 1573     :
0059 1574     :
0059 1575     :
0059 1576     :
0059 1577     :
0059 1578     :
0059 1579     :
0059 1580     :
0059 1581     :
0059 1582     :
0059 1583     :
0059 1584     :
0059 1585     :
0059 1586     :
0059 1587     :
0059 1588     :
0059 1589     :
0059 1590     :
0059 1591     :
0059 1592     :
0059 1593     :
0059 1594     :
0059 1595     :
0059 1596     :
0059 1597     :
0059 1598     :
0059 1599     :
0059 1600     :
0059 1601     :
0059 1602     :
0059 1603     :
0059 1604     :
0059 1605     :
0059 1606     :
0059 1607     :
0059 1608     :
0059 1609     :
0059 1610     :
0059 1611     :
0059 1612     :
0059 1613     :
0059 1614     :
0059 1615     :
0059 1616     :
0059 1617     :
0059 1618     :
0059 1619     :
0059 1620     :
0059 1621     :
0059 1622     :
0059 1623     :
0059 1624     :
0059 1625     :
0059 1626     :
0059 1627     :
0059 1628     :
0059 1629     :
0059 1630     :
0059 1631     :
0059 1632     :
0059 1633     :
0059 1634     :
0059 1635     :
0059 1636     :
0059 1637     :
0059 1638     :
0059 1639     :
0059 1640     :
0059 1641     :
0059 1642     :
0059 1643     :
0059 1644     :
0059 1645     :
0059 1646     :
0059 1647     :
0059 1648     :
0059 1649     :
0059 1650     :
0059 1651     :
0059 1652     :
0059 1653     :
0059 1654     :
0059 1655     :
0059 1656     :
0059 1657     :
0059 1658     :
0059 1659     :
0059 1660     :
0059 1661     :
0059 1662     :
0059 1663     :
0059 1664     :
0059 1665     :
0059 1666     :
0059 1667     :
0059 1668     :
0059 1669     :
0059 1670     :
0059 1671     :
0059 1672     :
0059 1673     :
0059 1674     :
0059 1675     :
0059 1676     :
0059 1677     :
0059 1678     :
0059 1679     :
0059 1680     :
0059 1681     :
0059 1682     :
0059 1683     :
0059 1684     :
0059 1685     :
0059 1686     :
0059 1687     :
0059 1688     :
0059 1689     :
0059 1690     :
0059 1691     :
0059 1692     :
0059 1693     :
0059 1694     :
0059 1695     :
0059 1696     :
0059 1697     :
0059 1698     :
0059 1699     :
0059 1700     :
0059 1701     :
0059 1702     :
0059 1703     :
0059 1704     :
0059 1705     :
0059 1706     :
0059 1707     :
0059 1708     :
0059 1709     :
0059 1710     :
0059 1711     :
0059 1712     :
0059 1713     :
0059 1714     :
0059 1715     :
0059 1716     :
0059 1717     :
0059 1718     :
0059 1719     :
0059 1720     :
0059 1721     :
0059 1722     :
0059 1723     :
0059 1724     :
0059 1725     :
0059 1726     :
0059 1727     :
0059 1728     :
0059 1729     :
0059 1730     :
0059 1731     :
0059 1732     :
0059 1733     :
0059 1734     :
0059 1735     :
0059 1736     :
0059 1737     :
0059 1738     :
0059 1739     :
0059 1740     :
0059 1741     :
0059 1742     :
0059 1743     :
0059 1744     :
0059 1745     :
0059 1746     :
0059 1747     :
0059 1748     :
0059 1749     :
0059 1750     :
0059 1751     :
0059 1752     :
0059 1753     :
0059 1754     :
0059 1755     :
0059 1756     :
0059 1757     :
0059 1758     :
0059 1759     :
0059 1760     :
0059 1761     :
0059 1762     :
0059 1763     :
0059 1764     :
0059 1765     :
0059 1766     :
0059 1767     :
0059 1768     :
0059 1769     :
0059 1770     :
0059 1771     :
0059 1772     :
0059 1773     :
0059 1774     :
0059 1775     :
0059 1776     :
0059 1777     :
0059 1778     :
0059 1779     :
0059 1780     :
0059 1781     :
0059 1782     :
0059 1783     :
0059 1784     :
0059 1785     :
0059 1786     :
0059 1787     :
0059 1788     :
0059 1789     :
0059 1790     :
0059 1791     :
0059 1792     :
0059 1793     :
0059 1794     :
0059 1795     :
0059 1796     :
0059 1797     :
0059 1798     :
0059 1799     :
0059 1800     :
0059 1801     :
0059 1802     :
0059 1803     :
0059 1804     :
0059 1805     :
0059 1806     :
0059 1807     :
0059 1808     :
0059 1809     :
0059 1810     :
0059 1811     :
0059 1812     :
0059 1813     :
0059 1814     :
0059 1815     :
0059 1816     :
0059 1817     :
0059 1818     :
0059 1819     :
0059 1820     :
0059 1821     :
0059 1822     :
0059 1823     :
0059 1824     :
0059 1825     :
0059 1826     :
0059 1827     :
0059 1828     :
0059 1829     :
0059 1830     :
0059 1831     :
0059 1832     :
0059 1833     :
0059 1834     :
0059 1835     :
0059 1836     :
0059 1837     :
0059 1838     :
0059 1839     :
0059 1840     :
0059 1841     :
0059 1842     :
0059 1843     :
0059 1844     :
0059 1845     :
0059 1846     :
0059 1847     :
0059 1848     :
0059 1849     :
0059 1850     :
0059 1851     :
0059 1852     :
0059 1853     :
0059 1854     :
0059 1855     :
0059 1856     :
0059 1857     :
0059 1858     :
0059 1859     :
0059 1860     :
0059 1861     :
0059 1862     :
0059 1863     :
0059 1864     :
0059 1865     :
0059 1866     :
0059 1867     :
0059 1868     :
0059 1869     :
0059 1870     :
0059 1871     :
0059 1872     :
0059 1873     :
0059 1874     :
0059 1875     :
0059 1876     :
0059 1877     :
0059 1878     :
0059 1879     :
0059 1880     :
0059 1881     :
0059 1882     :
0059 1883     :
0059 1884     :
0059 1885     :
0059 1886     :
0059 1887     :
0059 1888     :
0059 1889     :
0059 1890     :
0059 1891     :
0059 1892     :
0059 1893     :
005
```



```

E8 11 008F 1021 BRB 30$ ; Look at next adapter.
      0091 1022 40$:
      0091 1023
      0091 1024
      0091 1025 ; Store boot parameters.
      0091 1026
      0091 1031
      0095 1032 MOVL R2,RPBSL_ADPVIR(R9) ; Store VA of boot adapter space.
      0099 1033 MOVL R4,RPBSL_BOOTR1(R9) ; Store boot adapter nexus number.
      009F 1034 EXTZV #0,#13,= ; Get offset into UNIBUS/QBUS I/O page.
      009F 1035 RPBSL_CSRPHY(R9),R1
      00A6 1036 MOVAB <8*512>(R2)[R1],- ; Set VA of UNIBUS/QBUS registers.
      00A6 1037
      00A6 1038
      00A6 1039
      00A6 1040 ; R5/ general adapter type; index into "general" adapter arrays.
      00A6 1041 ; For each adapter -
      00A6 1042 ; Map the # of pages specified in ADAPDESC macro
      00A6 1043 ; JSB to initialization routine specified in ADAPDESC macro
      00A6 1044
      00A6 1045 MAP_NEXUS:
      00A6 1047 MOVAB <16*512>(R8),R8 ; Since no Qbus adapter space, point to
      00AB 1048 ; non-exist memory past Qbus I/O space.
      00AB 1050 MOVL R2,W^SBICONF[R4] ; Save VA of adapter space in SBICONF.
      00B1 1051 MOVZWL W^NUM_PAGES[R5],R1 ; Get number of pages to map.
      00B7 1052 BSBB MAP_PAGES ; Map the I/O pages.
      00B9 1053 MOVAL W^INIT_ROUTINES[R5],R1 ; Get address of initialization routine.
      00BF 1054 TSTL (R1) ; Initialization routine specified?
      00C1 1055 BEQL END_NEXUS ; Branch if none.
      00C3 1056 JSB @ (RT)[R1] ; Call initialization routine.
      00C7 1057 END_NEXUS:
      00C7 1058 INCL R4 ; Increment CONFREG and SBICONF index.
      00C9 1062 RSB ; Return, as only one nexus.
      00CA 1064

```

			00CA	1066		.SBTTL	CREATE_ARRAYS	
			00CA	1067				
			00CA	1068		CREATE_ARRAYS		
			00CA	1069				
			00CA	1070		Move the local CONFREG and SBICONF arrays into non-paged pool.		
			00CA	1071				
			00CA	1072		Inputs:		
			00CA	1073		R4 - Number of nexuses on the system.		
			00CA	1074		CONFREG and SBICONF have been initialized.		
			00CA	1075				
			00CA	1076		Outputs:		
			00CA	1077		R0 - R5 destroyed		
			00CA	1078		EXESGL_CONFREG points to a copy of the CONFREG array in non-paged pool		
			00CA	1079		MMG\$GL_SBICONF points to a copy of the SBICONF array in non-paged pool		
			00CA	1080		EXESGL_NUMNEXUS contains the number of nexuses on the system		
			00CA	1081				
			00CA	1082				
			00CA	1083		CREATE_ARRAYS:		
00000000'GF	54	DO	00CA	1084		MOVL	R4,G^EXESGL_NUMNEXUS	: Store number of nexuses on system.
51	OC A444	DE	00D1	1085		MOVAL	12(R4)[R4],R1	: Allocate n bytes for CONFREG plus
			00D6	1086				: 4n bytes for SBICONF + header
51	6144	DE	00D6	1087		MOVAL	(R1)[R4],R1	: Another 4n bytes for CONFREG.
	017D	30	00DA	1088		BSBW	ALONPAGD	: Get pool for CONFREG and SBICONF.
	82	7C	00DD	1089		CLRQ	(R2)+	: Clear out unused
82	82 51	BO	00DF	1090		MOVW	R1,(R2)+	: Set in size
00000000'GF	62	BO	00E2	1091		MOVW	#<DYN\$C CONF\$B>!DYN\$C-INIT,(R2)+	: Set type and subtype
51	6244	9E	00E7	1092		MOVAB	(R2),G^EXESGL_CONFREG	: Store address of system CONFREG.
00000000'GF	51	9E	00EE	1093		MOVAB	(R2)[R4],R1	: Two steps to CONFREG, 1st, SBICONF.
00000000'GF	6144	DO	00F2	1094		MOVL	R1,G^MMG\$GL_SBICONF	: Store address of system SBICONF.
	14	LE	00F9	1095		MOVAL	(R1)[R4],G^EXESGL_CONFREG	: And address of system CONFREG.
62	0020'CF	BB	0101	1096		PUSHR	#^M<R2,R4>	: Save pool address and nexus count.
	54	28	0103	1097		MOVCS	R4,W^CONFREG,(R2)	: Copy CONFREG to pool.
51	54	BA	0109	1098		POPR	#^M<R2,R4>	: Retrieve pool address and nexus count.
	7E	C5	010B	1099		MULL3	#4,R4,R1	: Number of bytes in SBICONF.
6244	0060'CF	DO	010F	1100		MOVL	R1,-(SP)	: Save, SBICONF size = CONFREG size
	51	28	0112	1101		MOVCS	R1,W^SBICONF,(R2)[R4]	: Copy SBICONF to pool.
63	0160'CF	DO	0119	1102		MOVL	(SP)+,R1	: Restore size of SBICONF and CONFREG.
		28	011C	1103		MOVCS	R1,W^CONFREG,(R3)	: Copy CONFREG to pool. R3 is output
			0122	1104				: from SBICONF MOVCS, so SBICONF and
			0122	1105				: CONFREG must be adjacent.
			0122	1106				
		05	0122	1107		RSB		

```

0123 1109 .SBTTL MAP_PAGES
0123 1110 :++
0123 1111 : INPUTS:
0123 1112 : R1/ Number of pages to map.
0123 1113 : R2/ VA of page to map.
0123 1114 : R3/ VA of system page table entry to be used.
0123 1115 : R8/ PFN of page(s) to map.
0123 1116 :
0123 1117 : OUTPUTS:
0123 1118 : R2,R3 updated; R1,R8 destroyed; all other registers preserved
0123 1119 :
0123 1120 :--
0123 1121 :
0123 1122 MAP_PAGES:
0123 1123
0123 1124 BISL3 #<PTESM_VALID!PTESC_KW>,R8,(R3)+
0128 1125 : Map a page.
0128 1126 INCL R8 : Next PFN.
0128 1127 MOVAB 512(R2),R2 : Next VA.
0132 1128 INCL G^BOO$GL_SPTFREL : Next free entry.
0138 1129 CMPL G^BOO$GL_SPTFREL, - : Check for no more system page
0143 1130 G^BOO$GL_SPTFREL : table entries.
0143 1131 BLEQ ERROR_HALT : Branch if out of SPTes.
0145 1132 SOBGTR R1,MAP_PAGES : Map another page.
0148 1133 RSB : All done.
0149 1134
0149 1135 ERROR_HALT:
0149 1136 MOVAB W^NOSPT,R1 : Set error message.
014E 1137 ERROR_HALT_1:
014E 1138 CLRL R11 : Indicate console terminal.
0150 1139 JSB G^EXE$OUTZSTRING : Output error message.
00 0156 1140 HALT : ***** FATAL ERROR *****

```



```

0157 1269 .SBTTL INISUBSPACE
0157 1270 :++
0157 1271 Map UNIBUS space; initialize UNIBUS ADP.
0157 1272
0157 1273 INPUTS:
0157 1274 R2 - VA of next free system page
0157 1275 R3 - VA of system page table entry to be used to map VA in R2
0157 1276 R4 - nexus identification number of this adapter
0157 1277 -8(R6) - PFN of this UNIBUS adapter's register space
0157 1278
0157 1279 OUTPUTS:
0157 1280 UNIBUS space is mapped.
0157 1281 INISUBADP is called to build an ADP block and initialize UNIBUS
0157 1282 adapter hardware.
0157 1283
0157 1284 :--
0157 1285
0157 1286 INISUBSPACE:
0157 1287
58 58 0160'CF44 DE 0157 1290 MOVAL W^CONFREGL[R4],R8 ; R8 => CONFREGL slot.
58 68 02 00 EF 015D 1291 EXTZV #0,#2,(R8),R8 ; Get UBA number.
58 58 58 09 78 0162 1292 ASHL #9,R8,R8 ; Position UB number.
0166 1295
0166 1304
0166 1309
0166 1314
0166 1319
0166 1325
58 00100000 8F 58 C3 0166 1327 SUBL3 R8,#<IOUV1$AL_QBOSP/^X200>,R8 ; Get PFN of Qbus I/O page.
016E 1328 ;
016E 1330 ;
016E 1331 MOVL #16,R1 ; Number of pages to map (UB/Qbus space).
51 10 D0 0171 1332 BSBW MAP_PAGES ; Map I/O pages.
FFAF 30
0174 1333 :
0174 1334 : Call adapter initialization routine.
0174 1335 :
0174 1336 : BSBW INISUBADP ; Init ADP block.
0174 1337 : RSB

```

```

0174 1339 .SBTTL INISUBADP - BUILD ADP AND INITIALIZE UBA
0174 1340
0174 1341 :+ INISUBADP ALLOCATES AND FILLS IN AN ADAPTER CONTROL BLOCK, INTERRUPT
0174 1342 DISPATCHER AND CONNECTS THEM TO THE PROPER SCB VECTORS. A CALL IS
0174 1343 THEN MADE TO UBASINITIAL TO INITIALIZE THE ADAPTER HARDWARE.
0174 1344
0174 1345 INPUT:
0174 1346 R4 - nexus identification number of this adapter
0174 1347 R11- offset from beginning of SCB to correct SCB page for this adapter
0174 1348 :-
0174 1349
0174 1350 INISUBADP:
0174 1351
0174 1352 PUSHF #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; SAVE R0-R8
0178 1353
0178 1354 : Allocate and initialize Adapter Control Block (ADP).
0178 1355
0178 1356 MOVZWL W^CPU ADPSIZE,R1 ; PICK UP LENGTH OF ADP
017D 1357 BSBW ALONPAGD ; ALLOCATE SPACE FOR ADP
0180 1358 MOVW R1,ADPSW_SIZE(R2) ; SET SIZE INTO ADP BLOCK
0184 1359 MOVW #DYN$C_ADP,- ; AND SET TYPE OF BLOCK
0188 1360 ADPSB_TYPE(R2)
0188 1361 MOVW #ATS_UBA,- ; SET TYPE OF ADAPTER
018C 1362 ADPSW_ADPTYPE(R2)
018C 1363 MOVL W^SBI[CONF[R4]],- ; SET VA OF CONFIGURATION REG
0192 1364 ADPSL_CSR(R2)
0192 1365 MOVW R4,ADPSW_TR(R2) ; SET TR NUMBER FOR ADAPTER
0196 1366
0196 1367 MOVAL ADPSL_DPQFL(R2),R0 ; ADDRESS OF DATA PATH WAIT QUEUE
019A 1368 MOVL R0,(R0) ; INIT QUEUE HEADER
019D 1369 MOVL R0,4(R0) ;
01A1 1370
01A1 1371 MOVAL ADPSL_MRQFL(R2),R0 ; ADDRESS OF MAP WAIT QUEUE
01A5 1372 MOVL R0,(R0) ; INIT QUEUE HEADER
01A8 1373 MOVL R0,4(R0) ;
01AC 1374 CLRL ADPSL_LINK(R2) ; ZAP ADAPTER CHAIN LINK
01AF 1375 BSBW ADPLINK ; LINK ADP TO END OF LIST
01B2 1376
01B2 1377 : Initialize adapter interrupt vectors in System Control Block.
01B2 1378
01B2 1379 MOVL G^EXE$GL_SCB,R8 ; GET SCB ADDRESS
01B9 1380
01B9 1387
01B9 1447
01B9 1507
01B9 1508
01B9 1536
01B9 1537
01B9 1539
01B9 1540 MOVAL ^X200(R8),- ; REMAINING ADP INIT FOR MICRO-VAX I:
01BD 1541 ADPSL_VECTOR(R2) ; ASSUME UBO
01BF 1542 ; VECTOR SPACE
01BF 1543 MOVW #^XE,ADPSW_DPBITMAP(R2) ; MARK DATAPATHS 1-3 AVAILABLE
01C3 1544 MOVAL G^UBASUNEXT+1,R3 ; GET ADDR OF UNEXP INT SERVICE
01CA 1545 ; (+1 MEANS HANDLE ON INT STACK)
01CA 1546 MOVAL W^UBASINT0+1,R4 ; SPECIAL CASE TO COUNT PASSIVE RELEASE
01CF 1547

```

```

50 10 A2 D0 01CF 1548 :
80 54 D0 01CF 1549 : INIT QBUS VECTORS TO UNEXPECTED INTERRUPT SERVICE
51 7F 8F 9A 01CF 1550 :
80 53 D0 01CF 1551 : MOVL ADP$L_VECTOR(R2),R0 : GET ADDRESS OF VECTORS
FA 51 F5 01D3 1552 : MOVL R4,(R0)+ : SPECIAL CASE FOR VECTOR 0
01D6 1553 : MOVZBL #<NUMUBAVEC-1>,R1 : REST OF VECTORS
01DA 1554 30$: MOVL R3,(R0)+ : FILL VECTOR WITH UNEXP INT
01DD 1555 : SOBGTR R1,30$ : FILL ALL VECTORS
01E0 1556
01E0 1558
01E0 1559
01E0 1601
01E0 1602
01E0 1604
01E0 1605 : All memory on the QBUS is main memory. There is no memory analogous
01E0 1606 : to UNIBUS memory.
01E0 1607
01E0 1608 : Now locate the memory controllers and build a list of the addresses
01E0 1609 : at which they are located. This list is used by the memory error logic
01E0 1610 : in machine-check. This information must be determined outside of machine-
01E0 1611 : check, since the machine-check code cannot cause another machine-check
01E0 1612 : without causing a cpu double-error halt.
01E0 1613
01E0 1614 : The list is a count of controllers, followed by the virtual addresses
01E0 1615 : that are the memory controller CSRs. Each MSV-11P has a single word CSR.
01E0 1616
01E0 1617 : .ENABLE LSB
53 00000000'GF D0 01E0 1618 : MOVL G^EXE$GL_SCB,R3 : Get SCB address.
04 A3 DD 01E7 1619 : PUSHL 4(R3) : Save current mcheck handler address.
50 SE D0 01EA 1620 : MOVL SP,R0 : Mark current stack position.
04 A3 20'AF DE 01ED 1621 : MOVAL B^MCHK_HANDLER,4(R3) : Connect temp mcheck handler.
01F2 1622
51 00000000'GF D0 01F2 1623 : MOVL G^MMG$GL_SBICONF,R1 : Get address of SBICONF array.
51 51 61 D0 01F9 1624 : MOVL (R1),R1 : Get VA of Qbus I/O space.
00002440 8F C0 01FC 1625 : ADDL #<012100+*X1000>,R1 : Offset to memory controller CSR(772100).
54 00000000'GF DE 0203 1626 : MOVAL G^EXE$AL_MEMCSRS,R4 : Get address of memory CSR count.
56 04 A4 DE 020A 1627 : MOVAL 4(R4),R6 : Get address of buffer for CSRs.
55 D4 020E 1628 : CLRL R5 : Initialize index.
0210 1629
6145 B5 0210 1630 50$: TSTW (R1)[R5] : Touch possible memory CSR.
64 D6 0213 1631 : INCL (R4) : Count number of error bits set.
86 6145 3E 0215 1632 : MOVAW (R1)[R5],(R6)+ : Save address of this CSR
F3 55 10 F2 0219 1633 60$: AOBLS #16,R5,50$ : Loop through all possible CSRs.
09 11 021D 1634 : BRB 70$ : Continue with common code.
021F 1635
: TEMPORARY MACHINE CHECK HANDLER
021F 1636
021F 1637
021F 1638
021F 1639 : .ALIGN LONG : Align machine-check vector.
0220 1640 MCHK_HANDLER: :
0220 1641
26 0F DA 0220 1642 : MTPR #^XF,#PRUV1$_MCESR : Clear machine-check state.
SE 50 D0 0223 1643 : MOVL R0,SP : Clean mcheck frame from stack.
F1 11 0226 1644 : BRB 60$ : Continue looking for memory CSRs.
0228 1645
04 A3 8ED0 0228 1646 70$: POPL 4(R3) : Restore mcheck handler address.
022C 1647

```



```

        56 62 D0 022C 1648 .DISABLE LSB
        51 D4 022C 1661 MOVL ADPSL_CSR(R2),R6 ; Pick up adapter pointer
0256 C2 51 B0 022F 1662 CLRL R1 ; Zero out number of UMR to disable
        0231 1686 MOVW R1,ADPSW_UMR_DIS(R2) ; Record number disabled
        0236 1700
        0236 1701 :: Initialize fields for the Qbus map register allocation. Since there
        0236 1702 :: are no map registers for the Micro-VAX I Qbus, initialize the data structures
        0236 1703 :: so that the standard allocate routine will just return an error.
        0236 1704 ::
64 A2 5C A2 01 D0 0236 1705 MOVL #1,ADPSL_MRACTMDRS(R2) ; 1 active map descriptor
        01F0 8F 51 A3 023A 1706 SUBW3 R1,#496,ADPSW_MRNREGARY(R2); for a range of 496 registers
        0241 1707 :: CLRL ADPSL_MRACTMDRS(R2) ; No active descriptors.
        0241 1708 :: CLRL ADPSW_MRNREGARY(R2) ; No registers to allocate,
015E C2 51 B0 0241 1710 MOVW R1,ADPSW_MRFREGARY(R2) ; starting at register zero.
        62 A2 01 AE 0246 1711 MNEGW #1,ADPSW_MRNFFENCE(R2) ; Also init "fences" which precede
015C C2 01 AE 024A 1712 MNEGW #1,ADPSW_MRFFENCE(R2) ; the two descriptor arrays.
        024F 1713 ::
        024F 1714 :: Initialize adapter hardware.
        024F 1715 ::
        54 62 D0 024F 1716 MOVL ADPSL_CSR(R2),R4 ; Get CSR address to init
        FDAB 30 0252 1717 BSBW UBASINITIAL ; And initialize adapter
        01FF 8F BA 0255 1718 POPR #^M<R0,R1,R2,R3,R4,R5,R6,R7,R8> ; Restore registers
        05 0259 1719 RSB ; Return
        025A 1720
        025A 1728

```

```

025A 1815      .SBTTL INISMBADP - BUILD ADP AND INITIALIZE MBA
025A 1816      .SBTTL INISDRADP - BUILD ADP AND INITIALIZE DR32
025A 1817      .SBTTL INISCIADP - BUILD ADP AND INITIALIZE CI
025A 1818      :+
025A 1819      : INISMBADP IS CALLED AFTER MAPPING THE REGISTERS FOR A MASSBUS ADAPTER.
025A 1820      : AN ADAPTER CONTROL BLOCK IS ALLOCATED AND FILLED. A CRB AND IDB ARE
025A 1821      : ALSO ALLOCATED AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED
025A 1822      : BY CALLING MBASINITIAL.
025A 1823      :
025A 1824      : INISDRADP IS CALLED AFTER MAPPING THE REGISTERS FOR THE DR32
025A 1825      : ADAPTER. THE ADAPTER CONTROL BLOCK, CRB, AND IDB ARE ALLOCATED
025A 1826      : AND INITIALIZED. THE ADAPTER HARDWARE IS THEN INITIALIZED BY
025A 1827      : CALLING DR$INITIAL.
025A 1828      :
025A 1829      : INISMBADP AND INISDRADP SHARE COMMON CODE AFTER THE TABLE OF ADAPTER
025A 1830      : SPECIFIC CONSTANTS IS SELECTED AND STORED IN R8.
025A 1831      :
025A 1832      : INPUT:
025A 1833      : R4 - nexus identification number of this adapter
025A 1834      : R11- offset from beginning of SCB to correct SCB page for this adapter
025A 1835      :
025A 1836      : OUTPUTS:
025A 1837      : ALL REGISTERS PRESERVED
025A 1838      : -
025A 1839      :
00000000'GF 17 025A 1840 ALONPAGD:JMP      G^INISALONONPAGED
0260 1841
0260 1842      .ENABL  LSB
0260 1843
0260 1844 INISDRADP:                                ; INITIALIZE DR32 DATA STRUCTURES
0260 1845
0260 1855 INISCIADP:                                ; INITIALIZE CI DATA STRUCTURES
0260 1856
0260 1857
0260 1867 INISMBADP:                                ; INIT MBA DATA STRUCTURES
0260 1868
0260 1869

```

INIADPUV1
V04-002

G 16

- ADAPTER INITIALIZATION FOR MICRO-VAX I 16-SEP-1984 01:04:35 VAX/VMS Macro V04-00
INISKDZ11 11-SEP-1984 16:29:18 [SYSLOA.SRC]INIADP.MAR;3

Page 23
(14)

```
0260 1997      .SBTTL INISKDZ11
0260 1998      :++
0260 1999      :
0260 2000      : INPUTS:
0260 2001      : R2 - VA of next free system page
0260 2002      : R3 - VA of system page table entry to be used to map VA in R2
0260 2003      : R4 - nexus identification number of this adapter
0260 2004      :
0260 2005      : OUTPUTS:
0260 2006      :
0260 2007      : --
0260 2008      :
0260 2009      : INISKDZ11:
0260 2010      :
05 0260 2029      RSB
```

; Return to caller.


```

0261 2031 .SBTTL INISCONSOLE, init data structures for console
0261 2032
0261 2033 ++
0261 2034 FUNCTIONAL DESCRIPTION:
0261 2035 This routine is executed only once, during system initialization.
0261 2036 It initializes the CRB and IDB for boot/console device.
0261 2037
0261 2038 This routine is called from INIT.
0261 2039
0261 2040 INPUTS:
0261 2041
0261 2042 R3 --> DISK [CLASS] DRIVER DDB
0261 2043 R4 --> DISK [CLASS] DRIVER DPT
0261 2044 R5 --> DISK [CLASS] DRIVER UCB
0261 2045 R6 --> RPB
0261 2046 R7 --> ADP FOR EITHER A REAL DISK OR A PORT
0261 2047 R9 --> PORT DRIVER DPT (IF PRESENT)
0261 2048 R10--> PORT DIRVER UCB (IF PRESENT)
0261 2049
0261 2050 --
0261 2051
0261 2052 INISCONSOLE::
0261 2053 .ENABL LSB
0261 2054
0261 2062
0261 2067
0261 2075
0261 2076
0261 2077 : NOW BUILD THE AUXILIARY DATA BLOCKS (CRB,IDB)
0261 2078
0261 2079 BLD_CRB:
0261 2080 MOVL ADPSL CRB(R7),R8 ; GET ADDRESS OF CRB IF IT EXISTS
0261 2081 CMPW #ATS_OBA,ADPSW_ADPTYPE(R7) ; IS THIS A UNIBUS ADAPTER?
0261 2082 BEQL FILL_CRB ; YES, ALLOCATE CRB
0261 2083 BRW 100$ ; NO, CRB/IDB ALREADY ALLOCATED
0261 2084
0261 2085 FILL_CRB:
0261 2086 JSB @#INISALLOC_CRB ; GO ALLOCATE AND SETUP CRB
0261 2087 MOVL #X9F163FBB,CRBSL_INTD(R2) ; SET PUSH# #M<R0,...,R5>
0261 2088 JSB @#0 INTO INTERRUPT DISPATCH
0261 2089 MOVL R7,CRBSL_INTD+VECSL_ADP(R2) ; SET POINTER TO ADP
0261 2090 MOVL R2,R8 ; SAVE CRB POINTER
0261 2091 MOVZWL #<IDB$C_LENGTH+<8*4>>,R1 ; SIZE TO ALLOCATE FOR IDB
0261 2092 JSB @#INISACONONPAGED ; ALLOCATE IDB
0261 2093 MOVW R1,IDB$W_SIZE(R2) ; SET SIZE OF IDB
0261 2094 MOVB #DYN$C_IDB,IDB$B_TYPE(R2) ; AND STRUCTURE TYPE CODE
0261 2095 MOVL R2,CRBSL_INTD+VECSL_IDB(R8) ; SET IDB INTO CRB
0261 2096
0261 2097
0261 2098
0261 2099
0261 2100
0261 2101
0261 2102
0261 2103
0261 2104
0261 2105
0261 2106
0261 2107
0261 2108
0261 2109
0261 2110
0261 2111
0261 2112
0261 2113
0261 2114 10$: MOVL RPB$SL_CSRVIR(R6), - ; SAVE BOOT DEVICE CSR ADDRESS
0261 2115 IDB$SL_CSR(R2) ; IN INTERRUPT DISPATCH BLOCK
0261 2116 CMPB #BTD$R_UDA,- ; LOW ORDER BYTE OF ORIGINAL R0 TELLS
0261 2117 RPB$B_DEVTP(R6) ; BOOT DEVICE TYPE.
0261 2118 BNEQ 20$ ; IF NOT BOOTING FROM A UDA BRANCH
0261 2119 ; AROUND.
0261 2120 MOVL RPB$SL_CSRVIR(R6), - ; COPY VIRTUAL ADDRESS OF UDA PORT CSR
0261 2121 @#BOO$GB_SYSTEMID ; TO LOW ORDER LONGWORD OF SYSTEMID

```

14	A2	57	D0	02AC	2122	20\$:	MOVL	R7, IDBSL ADP(R2)	:	POINT IDB TO ADP
50	1E	A6	3C	02AC	2123		MOVZWL	RPBSW_R00BVEC(R6), R0	:	GET USER SPECIFIED VECTOR
		0A	12	02B4	2125		BNEQ	30\$:	BRANCH IF VECTOR SPECIFIED
50	66	A6	9A	02B6	2126		MOVZBL	RPBSB DEVTYP(R6), R0	:	ELSE GET DEVICE TYPE CODE
50	FFFE	CF40	3C	02BA	2127		MOVZWL	W^BOOTVECTOR-2[R0], R0	:	GET DEFAULT INTERRUPT VECTOR
50	10	B740	9E	02C0	2128	30\$:	MOVAB	@ADPSL_VECTOR(R7)[R0], R0	:	COMPUTE ADDRESS OF VECTOR
60	26	A8	9E	02C5	2129		MOVAB	CRBSL_INTD+2(R8), (R0)	:	SET ADDR OF INTERRUPT VECTOR
				02C9	2130				:	
		60	D7	02C9	2133		DECL	(R0)	:	BACK TWO BYTES TO PUSHR, +1 TO
				02CB	2136				:	
				02CB	2137	100\$:			:	
			05	02CB	2138		RSB		:	RETURN
				02CC	2139		.DISABLE LSB		:	

```

02CC 2141 .SBTTL EXESINI_TIMWAIT - COMPUTE CORRECT TIMEWAIT LOOP VALUES
02CC 2142
02CC 2143 ++
02CC 2144 FUNCTIONAL DESCRIPTION:
02CC 2145 EXESINI_TIMWAIT initializes EXESGL_TENUSEC and EXESGL_UBDELAY, cells used
02CC 2146 in the time-wait macros. The first data cell, EXESGL_TENUSEC, is the number
02CC 2147 of times the following loop will be executed in ten u-seconds. This is
02CC 2148 done once here to calibrate the loop instead of reading the processor clock.
02CC 2149 The resulting number is used in the system macros TIMEWAIT and TIMEDWAIT.
02CC 2150
02CC 2151 The first step is to initialize EXESGL_UBDELAY. If the bit test instruction
02CC 2152 in the TIMEWAIT macro is executed too rapidly in a loop, it can saturate the
02CC 2153 Unibus. EXESGL_UBDELAY is used to introduce a 3 microsecond delay loop into
02CC 2154 the TIMEWAIT bit test loop.
02CC 2155
02CC 2156 This routine is called only once, from INIT.
02CC 2157
02CC 2158 INPUT PARAMETERS:
02CC 2159
02CC 2160 NONE
02CC 2161
02CC 2162 IMPLICIT INPUTS:
02CC 2163
02CC 2164 Time-of-day processor clock.
02CC 2165 Interval timers.
02CC 2166
02CC 2167 OUTPUT PARAMETERS:
02CC 2168
02CC 2169 R0 - Destroyed.
02CC 2170
02CC 2171 IMPLICIT OUTPUTS:
02CC 2172
02CC 2173 EXESGL_TENUSEC - set to appropriate value to make TIMEWAIT and TIMEDWAIT
02CC 2174 macros loop for 10 micro-seconds.
02CC 2175
02CC 2176 EXESGL_UBDELAY - set to appropriate value to make TIMEWAIT and TIMEDWAIT
02CC 2177 macros loop for 3 micro-seconds in the unibus delay
02CC 2178 loop.
02CC 2179
02CC 2180 --
02CC 2181
02CC 2182 EXESINI_TIMWAIT::
02CC 2294 MOVZBL #1,G*EXESGL_UBDELAY ; Initialize time-wait data cells
02CC 2295 MOVZBL #1,G*EXESGL_TENUSEC ; Set UV1 value same as 11/730
02DA 2296 RSB ; Set UV1 value same as 11/730
02CC 2297 ; Return

```



```

02DB 2299 .SBTTL EXESINIT_TODR - SET SYSTEM TIME TO CORRECT VALUE AT STARTUP
02DB 2300 :++
02DB 2301 : FUNCTIONAL DESCRIPTION:
02DB 2302 :
02DB 2303 : EXESINIT_TODR SOLICITS THE CORRECT TIME FROM THE OPERATOR IF NECESSARY,
02DB 2304 : CONVERTS THE ASCII RESPONSE TO BINARY FORMAT AND CALLS AN INTERNAL
02DB 2305 : ENTRY POINT OF THE $SETIME SYSTEM SERVICE TO SET THE NEW SYSTEM TIME
02DB 2306 : IN MEMORY WITHOUT MODIFYING THE CONTENTS OF THE SYSTEM DISK.
02DB 2307 :
02DB 2308 : IF THE TIME WOULD NORMALLY BE SOLICITED FROM AN OPERATOR, BECAUSE
02DB 2309 : THE HARDWARE TIME OF YEAR CLOCK IS ZERO, THEN THE SYSGEN PARAMETER
02DB 2310 : "TPWAIT" IS CHECKED. IF IT IS ZERO, THEN IT IS ASSUMED THAT NO
02DB 2311 : OPERATOR IS PRESENT AND THE SYSTEM IS BOOTED USING THE LAST TIME
02DB 2312 : RECORDED IN THE SYSTEM IMAGE. IF THE PARAMETER IS NON ZERO THEN
02DB 2313 : THAT TIME IS USED AS THE MAXIMUM TIME TO WAIT BEFOR ASSUMING THAT
02DB 2314 : THERE IS NO OPERATOR AND BOOTING ANY WAY. IF THE PARAMETER IS
02DB 2315 : NEGATIVE, THE SYSTEM WILL WAIT FOREVER.
02DB 2316 :
02DB 2317 : THIS ROUTINE IS CALLED ONLY ONCE, FROM SYSINIT OR STASYSGEN.
02DB 2318 :
02DB 2319 : INPUT PARAMETERS:
02DB 2320 :
02DB 2321 : NONE
02DB 2322 :
02DB 2323 : IMPLICIT INPUTS:
02DB 2324 :
02DB 2325 : TIME-OF-DAY PROCESSOR CLOCK.
02DB 2326 :
02DB 2327 : OUTPUT PARAMETERS:
02DB 2328 :
02DB 2329 : R0,R1 - DESTROYED
02DB 2330 :
02DB 2331 : IMPLICIT OUTPUTS:
02DB 2332 :
02DB 2333 : EXESGQ_SYSTIME - SET TO CURRENT TIME IN 100 NANOSECOND UNITS SINCE
02DB 2334 : 17-NOV-1858 00:00:00.
02DB 2335 :
02DB 2336 : --
02DB 2337 :
02DB 2338 :
02DB 2339 : Stack storage offsets:
02DB 2340 :
02DB 2341 : TCHAN = ^X00 : CHANNEL FOR TERMINAL (LONGWORD)
02DB 2342 : TTNAME = ^X04 : STRING DESCRIPTOR FOR OPERATOR'S TERM
02DB 2343 : TMPDESC = ^X0C : TEMPORARY STRING DESCRIPTOR (QUADWORD)
02DB 2344 : INTIME = ^X14 : INPUT TIME VALUE (QUADWORD)
02DB 2345 : LINBUF = ^X1C : INPUT LINE BUFFER (5 LONGWORDS)
02DB 2346 : LINBUFSIZ = ^X14 : (LENGTH OF LINE BUFFER IN BYTES)
02DB 2347 :
02DB 2348 :
02DB 2349 : PURE DATA
02DB 2350 :
02DB 2351 : TERM_NAMADR:
02DB 2352 : .ASCII \OPAO\ : DEVICE NAME FOR OPERATOR'S TERMINAL
02DB 2353 : TERM_NAMSIZ = - TERM_NAMADR :
02DB 2354 : TIMERR: .ASCII \invalid date/time\ :
02EB

```

30 41 50 4F
00000004
0000000C
00000014
0000001C
00000014

74 61 64 20 64 69 6C 61 76 6E 69 00
65 6D 69 74 2F 65

```

54 4E 45 20 45 53 41 45 4C 50 0A 0D 33' 02DF1 2355 TIMEPROMPT:
20 44 4E 41 20 45 54 41 44 20 52 45 02F1 2356 .BYTE NPROMPT
4D 4D 4D 2D 44 44 28 20 45 4D 49 54 02F2 2357 .ASCII <13><10>/PLEASE ENTER DATE AND TIME (DD-MMM-YYYY HH:MM) /
4D 4D 3A 48 48 20 20 59 59 59 59 2D 030A
00000033 0316
0322
0323 2358 NPROMPT=.-TIMEPROMPT-1
0324 2359
0325 2360
0326 2361 EXESINIT TODR: ; SET CORRECT TIME
0327 2362 .ENABLE LSB
0328 2363 PUSHR #M<R2,R3,R4,R5,R6,R8,R9,R10> ; SAVE REGISTERS
0329 2364 SUBL #4*12,$P ; SCRATCH STORAGE
032C 2365 MOVL SP,R6 ; SAVE ADDRESS OF SCRATCH STORAGE
032F 2366 MOVZBL #TERM NAMSIZ,TTNAME(R6) ; SET SIZE OF OPERATOR'S TERM NAME AND
0333 2367 MOVAB W*TERM NAMADR,TTNAME+4(R6) ; PIC ADDRESS INTO TERM NAME DESC
0339 2368 BBS S*EXESV_SETTIME,G*EXESGL_FLAGS,READTIME ; BR TO SOLICIT TIME
0341 2369
0341 2370
0341 2374
0341 2378
0341 2382
0341 2386
0341 2394
0343 2396 58: BRB READTIME ; ON MICROVAX 1, ALWAYS SOLICIT TIME
0346 2397 CLRQ INTIME(R6) ; NULL ARGUMENT FOR EXESSETIME_INT
0349 2398 BRW 2008 ; RETURN TO CALLER
0349 2399 READTIME: ; SOLICIT TIME
0349 2400 CLRL R9 ; CLEAR A FLAG
034B 2401 CVTBL G*SGN$GW_TPWAIT,R8 ; PICK UP TIMEOUT WAIT INTERVAL
0352 2402 BGTR 88 ; POSITIVE, WAIT THAT PERIOD ONCE
0354 2403 BLSS 78 ; NEGATIVE IS WAIT FOREVER
0356 2404 68:
0356 2408
0356 2412
0356 2416
0356 2420
0356 2424
0356 2428
0356 2430
035E 2431
0361 2433
0361 2434 78: MOVL #20,R8 ; STARTING WAIT
0364 2435 INCL R9 ; NEGATIVE - WAIT FOREVER
0366 2436 88: $ASSIGN S TTNAME(R6),TTCHAN(R6) ; AND ASSIGN TO INPUT DEVICE
0374 2437 BLBC -R0,68 ; ERROR - FALL BACK TO STORED TIME
0377 2438 108: MOVAB W*TIMEPROMPT,R2 ; GET ADDRESS OF PROMPT STRING
037C 2439 MOVZBL (R2)+,R3 ; AND LENGTH
037F 2440 $QIOW,S #0 W*TTCHAN(R6) - ; PROMPT AND READ TIME
037F 2441 #<108 READPROMPT!IOSM_PURGE!IOSM_TIMED!IOSM_CVTLOW>,- ;
037F 2442 TMPDESC(R6) ; I/O STATUS BLOCK, NO ASI OR PARAM
037F 2443 LINBUF(R6),#LINBUFSIZ,- ; BUFFER ADDRESS AND SIZE
037F 2444 R8,#0,- ; TIME OUT
037F 2445 R2,R3 ; PROMPT ADDRESS AND SIZE
03A4 2446 BLBC R0,68 ; ERROR - FALL BACK TO STORED TIME
14 A6 00000000'GF 7D 0356 2430 MOVQ G*EXESGQ_TODCBASE,INTIME(R6) ; USE LAST KNOWN SYSTEM TIME
00AD 31 035E 2431 BRW 2008 ; IF THE USER REQUESTS NO PROMPTING
58 14 D0 0361 2433 78: MOVL #20,R8 ; STARTING WAIT
59 D6 0364 2435 INCL R9 ; NEGATIVE - WAIT FOREVER
DF 50 E9 0366 2436 88: $ASSIGN S TTNAME(R6),TTCHAN(R6) ; AND ASSIGN TO INPUT DEVICE
52 FF76 CF 9E 0374 2437 BLBC -R0,68 ; ERROR - FALL BACK TO STORED TIME
53 82 9A 0377 2438 108: MOVAB W*TIMEPROMPT,R2 ; GET ADDRESS OF PROMPT STRING
037C 2439 MOVZBL (R2)+,R3 ; AND LENGTH
037F 2440 $QIOW,S #0 W*TTCHAN(R6) - ; PROMPT AND READ TIME
037F 2441 #<108 READPROMPT!IOSM_PURGE!IOSM_TIMED!IOSM_CVTLOW>,- ;
037F 2442 TMPDESC(R6) ; I/O STATUS BLOCK, NO ASI OR PARAM
037F 2443 LINBUF(R6),#LINBUFSIZ,- ; BUFFER ADDRESS AND SIZE
037F 2444 R8,#0,- ; TIME OUT
037F 2445 R2,R3 ; PROMPT ADDRESS AND SIZE
AF 50 E9 03A4 2446 BLBC R0,68 ; ERROR - FALL BACK TO STORED TIME

```

54	OC A6	7D	03A7	2447	MOVQ	TMPDESC(R6),R4	:	GET COMPLETION STATUS
	OD 54	E8	03AB	2448	BLBS	R4,20\$:	CONTINUE IF SUCCESSFUL READ
	A5 59	E9	03AE	2449	BLBC	R9,6\$:	FAILED ON ONE-TIME READ, RETURN
58	01 A848	9E	03B1	2450	MOVAB	1(R8)[R8],P8	:	(2 * TIMEOUT) + 1
	58 58	3C	03B6	2451	MOVZWL	R8,R8	:	BOUND TIMEOUT
	BC	11	03B9	2452	BRB	10\$:	TRY AGAIN FOR TIME
			03BB	2453			:	SOMETHING WAS INPUT
OC A6	OE A6	3C	03BB	2454	MOVZWL	TMPDESC+2(R6),TMPDESC(R6)	:	FORM DESCRIPTOR FOR BUFFER
10 A6	1C A6	9E	03C0	2455	MOVAB	LINBUF(R6),TMPDESC+4(R6)	:	SET DESCRIPTOR ADDRESS
			03C5	2456	\$BINTIM_S	TMPDESC(R6),INTIME(R6)	:	CONVERT TO BINARY TIME
	05 50	E9	03D2	2457	BLBC	R0,89\$:	INVALID TIME
	18 A6	D5	03D5	2458	TSTL	INTIME+4(R6)	:	CHECK FOR DELTA TIME
	2A	14	03D8	2459	BGTR	100\$:	BRANCH IF NOT - OK
			03DA	2460			:	INVALID TIME VALUE INPUT
52	FF01 CF	9E	03DA	2461	MOVAB	W*TIMERR,R2	:	ADDRESS OF ERROR MESSAGE
	53 82	9A	03DF	2462	MOVZBL	(R2)+,R3	:	GET STRING LENGTH
			03E2	2463	\$QIOW_S	#0,TTCHAN(R6),-	:	GIVE ERROR MESSAGE
			03E2	2464		#10\$,WRITEVBLK,-	:	
			03E2	2465			:	NO I/O STATUS,AST OR AST PARAM
			03E2	2466		{R2},R3,-	:	BUFFER ADDRESS, LENGTH
			03E2	2467		#0,#32	:	SET CARRIAGE CONTROL TO CR/LF
	FF73	31	0401	2468	BRW	10\$:	AND TRY AGAIN
			0404	2469			:	EXIT
			0404	2470	\$DASSGN_S	TTCHAN(R6)	:	DE-ASSIGN TERMINAL CHANNEL
	14 A6	7F	040E	2471	PUSHAQ	INTIME(R6)	:	SET NEW SYSTEM TIME
00000000'GF	01	FB	0411	2472	CALLS	#1,G*EXES\$SETIME INT	:	USE TODR CLOCK TO SET SYSTEM TIME
00000000'GF	00000000'GF	7D	0418	2473	MOVQ	G*EXES\$GQ_TODCBASE,G*EXES\$GQ_BOOTTIME	:	SAVE BOOT TIME
	5E 30	C0	0423	2474	ADDL	#12*4,SP	:	CLEAN OFF SCRATCH STORAGE
	077C 8F	BA	0426	2475	POPR	#*M<R2,R3,R4,R5,R6,R8,R9,R10>	:	RESTORE REGISTERS
			042A	2476			:	
			042A	2477			:	
			042A	2478			:	
			042A	2479			:	
			042A	2480			:	
			042A	2481			:	
			042A	2482			:	
			042A	2483			:	
			042A	2484			:	

20\$:

89\$:

100\$:

200\$:

RSB

.DISABLE LSB

*** This goes in if another piece of
*** initialization code is added that
*** is executed after EXESINI_TIMWAIT.

Fall through into the deallocate logic.


```

042A 2486 DEAL_INIT_CODE: ; DEALLOCATE THE INITIALIZATION CODE
042A 2487 :
042A 2488 : It is the duty of the last-executed, loadable initialization
042A 2489 : routine to make itself and all other such routines disappear, i.e.,
042A 2490 : release the space they occupy to non-paged pool. Each routine's vector
042A 2491 : must be disconnected, e.g., be made to point to the symbol, EXES$LOAD_ERROR.
042A 2492 :
042A 2493 : NOTE: This means that new initialization routines should be added
042A 2494 : to this module in a particular order, not necessarily at the
042A 2495 : end of the module!
042A 2496 :
042A 2497 :
042A 2498 .ENABLE LSB
7E 52 7D 042A 2498 MOVQ R2,-(SP) ; Save some registers
042D 2499
042D 2500 :
042D 2501 : First find the vectors that point to these initialization routines
042D 2502 : and reset them to point to EXES$LOAD_ERROR.
042D 2503 :
042D 2504 :
51 50 0000'CF 9E 042D 2504 MOVAB W^SYSL$BEGIN,R0 ; Compute bounds of releasable piece:
50 00000000'8F C1 0432 2505 ADDL3 #<STAY_HEADER-SYSL$BEGIN>,R0,R1 ; starting and ending addresses.
52 00000000'GF 9E 043A 2506 MOVAB G^EXES$LOAD_ERROR,R2 ; Get starting address of vectors.
53 00000000'GF 9E 0441 2507 MOVAB G^EXES$LOAD_ERROR,R3 ; Get end of vectors.
9F17 8F 62 B1 0448 2508 10$: CMPW (R2),#^X9FT7 ; Is this JMP @#?
1B 13 044D 2509 BEQL 30$ ; Br if yes, skip past it.
80 8F 03 A2 91 044F 2510 CMPB 3(R2),#^X80 ; Is this a system space address
16 12 0454 2511 BNEQ 40$ ; Br if no, assume it's a HALT instr.
50 62 D1 0456 2512 CMPL (R2),R0 ; Is address before the releasable
OC 1F 0459 2513 BLSSU 20$ ; piece of memory? Br on yes.
51 62 D1 045B 2514 CMPL (R2),R1 ; Is address after the releasable
07 1A 045E 2515 BGTRU 20$ ; piece of memory? Br on yes.
62 00000000'GF 9E 0460 2516 MOVAB G^EXES$LOAD_ERROR,(R2) ; Reset this vector.
52 02 C0 0467 2517 20$: ADDL #2,R2 ; Point past this vector.
52 D6 046A 2518 30$: INCL R2 ; Come here to point past JMP @#.
52 D6 046C 2519 40$: INCL R2 ; Come here to point past HALT.
53 52 D1 046E 2520 CMPL R2,R3 ; Past the end of the vectors?
D5 1F 0471 2521 BLSSU 10$ ; Keep searching vectors.
0473 2522 :
0473 2523 : Now release the memory to non-paged pool.
0473 2524 :
50 0000'CF 9E 0473 2525 MOVAB W^SYSL$BEGIN,R0 ; Point to start of module
51 0000'8F 3C 0478 2526 MOVZWL #<STAY_HEADER-SYSL$BEGIN>,R1 ; Length to vaporize
FB8C' 31 047D 2527 BRW 50$ ; Br to code that is not released.
0480 2528
00000000 2529 .PSECT $$$INIT__END,PAGE ; 'PAGE' SINCE 16-BYTE ALIGN IS NOT
0000 2530
0000 2531 STAY_HEADER:
00000000 00000000 0000 2532 .LONG 0,0
0000' 0008 2533 .WORD <SYSL$END-STAY_HEADER>
62 000A 2534 .BYTE DYN$C_LOADCODE
00 000B 2535 .BYTE 0
000C 2536
00000000'9F 16 000C 2537 50$: JSB @#EXES$DEANONPGDSIZ ; Just the smile on the Cheshire cat
52 8E 7D 0012 2538 MOVQ (SP)+,R2 ; Restore
05 0015 2539 RSB ; Return.
0016 2540
0016 2541 .DISABLE LSB
0016 2542 .END

```

INIADPUV1
Symbol table

C 1

- ADAPTER INITIALIZATION FOR MICRO-VAX I 16-SEP-1984 01:04:35 VAX/VMS Macro V04-00
11-SEP-1984 16:29:18 [SYSLOA.SRC]INIADP.MAR;3

Page 31
(17)

\$\$\$VMSDEFINED	= 00000001			EXES\$DEANONPGDSIZ	*****	X	0A
\$ST1	= 00000001			EXESGL_CONFREG	*****	X	09
ADAPTERS	= 00000000	R	02	EXESGL_CONFREGL	*****	X	09
ADPSB_TYPE	= 0000000A			EXESGL_FLAGS	*****	X	09
ADPSC_UBAADPLEN	= 00000258			EYESGL_NUMNEXUS	*****	X	09
ADPSL_CRB	= 00000010			EXESGL_RPB	*****	X	09
ADPSL_CSR	= 00000000			EXESGL_SCB	*****	X	09
ADPSL_DPQFL	= 00000014			EXESGL_TENUSEC	*****	X	09
ADPSL_LINK	= 00000004			EXESGL_UBDELAY	*****	X	09
ADPSL_MRACTMDRS	= 0000005C			EXESGQ_BOOTTIME	*****	X	09
ADPSL_MRQFL	= 00000030			EXESGQ_TODCBASE	*****	X	09
ADPSL_VECTOR	= 00000010			EXESINIT TODR	00000325	RG	09
ADPSW_ADPTYPE	= 0000000E			EXESINI TIMWAIT	000002CC	RG	09
ADPSW_DPBITHAP	= 00000060			EXESLOAD ERROR	*****	X	09
ADPSW_MRFENCE	= 0000015C			EXESOUTZSTRING	*****	X	09
ADPSW_MRFREGARY	= 0000015E			EXESSETIME INT	*****	X	09
ADPSW_MRNFBCE	= 00000062			EXESV SETTIME	*****	X	09
ADPSW_MRNREGARY	= 00000064			FILL CRB	0000026E	R	09
ADPSW_SIZE	= 00000008			GET_GEN TYPE	0000006B	R	09
ADPSW_TR	= 0000000C			GET TYPE	00000068	R	09
ADPSW_UMR_DIS	= 00000256			IDB\$B_TYPE	= 0000000A		
ADPLINK	*****	X	09	IDB\$C_LENGTH	= 00000038		
ALONPAGD	0000025A	R	09	IDB\$L_ADP	= 00000014		
AT\$ UBA	= 00000001			IDB\$L_CSR	= 00000000		
BI_BUS_CODE	= 80000000			IDB\$W_SIZE	= 00000008		
BI_CPU	= 00000000			INISALCLOC CRB	*****	X	09
BI_CSR_LEN	= 00000002			INISALONONPAGED	*****	X	09
BI-LIKE	= 00000000			INISCIADP	00000260	R	09
BLD CRB	00000261	R	09	INISCONSOLE	00000261	RG	09
BOO\$GB_SYSTEMID	*****	X	09	INISDRADP	00000260	R	09
BOO\$GL_SPTFREN	*****	X	09	INISDIOMAP	00000000	RG	09
BOO\$GL_SPTFREL	*****	X	09	INISKDZ11	00000260	R	09
BOCTVECTOR	00000000	R	08	INISMBADP	00000260	R	09
BTDSK_UDA	= 00000011			INISMPMADP	*****	X	06
BUS_CODE_OFFSET	= FFFFFFFC			INISUBADP	00000174	R	09
BUS_CSR_LEN	00000004	R	08	INISUBSPACE	00000157	R	09
CONFIG_IOSPACE	00000059	R	09	INIT ROUTINES	00000000	R	06
CONFREG	00000020	R	08	INTIME	= 00000014		
CONFREGL	00000160	R	08	IOSM_CVTLOW	*****	X	09
CPU_ADPSIZE	0000000D	R	08	IOSM_PURGE	*****	X	09
CPU_TYPE	= 00000007			IOSM-TIMED	*****	X	09
CR	= 0000000D			IOS_READPROMPT	*****	X	09
CRRL INTD	= 00000024			IOS_WRITEVBLK	*****	X	09
CREATE ARRAYS	000000CA	R	09	IOU\$ISAL_QBOSP	= 20000000		
CSR_LEN_OFFSET	= FFFFFFFB			LF	= 0000000A		
DEAL_INIT_CODE	0000042A	R	09	LINBUF	= 0000001C		
DIRECT_VEC_NODE_CNT	00000009	R	08	LINBUFSIZ	= 00000014		
DYN\$C_ADP	= 00000001			MAP_NEXUS	000000A6	R	09
DYN\$C_CONF	= 00000007			MAP-PAGES	00000123	R	09
DYN\$C_IDB	= 00000009			MAXNEXUS	= 00000040		
DYN\$C_INIT	= 00000063			MCHK_HANDLER	00000220	R	09
DYN\$C_LOADCODE	= 00000062			MMG\$GL_SBICONF	*****	X	09
END NEXUS	000000C7	R	09	MMG\$GL_SPTBASE	*****	X	09
ERROR_HALT	00000149	R	09	NDTS_B0A	= 80000102		
ERROR_HALT_1	0000014E	R	09	NDTS-CI	= 00000038		
EXESAL_LOADEC	*****	X	09	NDTS-DR32	= 00000030		
EXESAL_MEMCSRS	*****	X	09	NDTS_KDZ11	= 80000105		

INIADPUV1
Symbol table

D 1

- ADAPTER INITIALIZATION FOR MICRO-VAX I 16-SEP-1984 01:04:35 VAX/VMS Macro V04-00
11-SEP-1984 16:29:18 [SYSLOA.SRC]INIADP.MAR;3

Page 32
(17)

```

NDTS_MB = 00000020
NDTS_MEM1664NI = 00000012
NDTS_MEM16I = 00000011
NDTS_MEM16NI = 00000010
NDTS_MEM256EIL = 00000071
NDTS_MEM256EIU = 00000073
NDTS_MEM256I = 00000074
NDTS_MEM256NIL = 00000070
NDTS_MEM256NIU = 00000072
NDTS_MEM4I = 00000009
NDTS_MEM4NI = 00000008
NDTS_MEM64EIL = 00000069
NDTS_MEM64EIU = 00000068
NDTS_MEM64I = 0000006C
NDTS_MEM64NIL = 00000068
NDTS_MEM64NIU = 0000006A
NDTS_MPM0 = 00000040
NDTS_MPM1 = 00000041
NDTS_MPM2 = 00000042
NDTS_MPM3 = 00000043
NDTS_SCORMEM = 80000001
NDTS_UB0 = 00000028
NDTS_UB1 = 00000029
NDTS_UB2 = 0000002A
NDTS_UB3 = 0000002B
NEXUSDESC = 00000014 R 08
NOSPT = 00000260 R 08
NPROMPT = 00000033
NUMUBAVEC = 00000080
NUM_PAGES = 00000000 R 04
NXT_NEXUS = 00000065 R 09
PA = 20000000
PR$_SID_TYP730 = 00000003
PR$_SID_TYP750 = 00000002
PR$_SID_TYP780 = 00000001
PR$_SID_TYP790 = 00000004
PR$_SID_TYP8NN = 00000006
PR$_SID_TYP8SS = 00000005
PR$_SID_TYPUV1 = 00000007
PRUVIS_MCESR = 00000026
PTESC_RW = 10000000
PTESM_VALID = 80000000
READTIME = 00000349 R 09
RPB$B_DEVTYP = 00000066
RPB$B_ADPVIR = 00000060
RPB$B_BOOTR1 = 00000020
RPB$B_CSRPHY = 00000054
RPB$B_CSRVIR = 00000058
RPB$W_ROUBVEC = 0000001E
SBICONF = 00000060 R 08
SBI_BUS_CODE = 00000000
SBI_CPU = 00000000
SBI_CSR_LEN = 00000001
SBI_LIKE = 00000001
SGN$GW_TPWAIT = ***** X 09
STAY_HEADER = 00000000 R 0A
SW_BDS_CODE = 00000005 R 08

```

```

SYSS$ASSIGN ***** GX 09
SYSS$BINTIM ***** GX 09
SYSS$DASSGN ***** GX 09
SYSS$QIOW ***** GX 09
SYSL$BEGIN ***** X 09
SYSL$END ***** X 0A
TERM_NAMADR = 000002DB R 09
TERM_NAMSIZ = 00000004
TIMEPROMPT = 000002F1 R 09
TIMERR = 000002DF R 09
TMPDESC = 0000000C
TTCHAN = 00000000
TTNAME = 00000004
UBA$INITIAL ***** X 09
UBA$INTO ***** X 09
UBA$UNEXINT ***** X 09
VASM_SYSTEM = 80000000
VEC$C_AD = 00000014
VEC$L_IDB = 00000008

```


+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes																	
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE							
\$ABSS	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE							
\$\$\$INIT\$DATA0	00000074 (116.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE							
\$\$\$INIT\$DATA1	00000000 (0.)	03 (3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE							
\$\$\$INIT\$DATA2	0000003A (58.)	04 (4.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE							
\$\$\$INIT\$DATA3	00000000 (0.)	05 (5.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE							
\$\$\$INIT\$DATA4	00000074 (116.)	06 (6.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE							
\$\$\$INIT\$DATA5	00000000 (0.)	07 (7.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE							
\$\$\$INIT\$DATA	00000289 (649.)	08 (8.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG							
\$\$\$INIT\$CODE	00000480 (1152.)	09 (9.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	QUAD							
\$\$\$INIT__END	00000016 (22.)	0A (10.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	PAGE							

+-----+
! Performance indicators !
+-----+

Phase	Page faults	CPU Time	Elapsed Time
Initialization	35	00:00:00.06	00:00:01.64
Command processing	141	00:00:00.45	00:00:03.20
Pass 1	507	00:00:12.82	00:00:47.78
Symbol table sort	0	00:00:01.65	00:00:06.88
Pass 2	234	00:00:03.78	00:00:17.39
Symbol table output	24	00:00:00.13	00:00:00.82
Psect synopsis output	4	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	947	00:00:18.92	00:01:17.74

The working set limit was 2100 pages.
132724 bytes (260 pages) of virtual memory were used to buffer the intermediate code.
There were 90 pages of symbol table space allocated to hold 1600 non-local and 24 local symbols.
2546 source lines were read in Pass 1, producing 36 object records in Pass 2.
42 pages of virtual memory were used to define 40 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	19
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	33

1745 GETS were required to define 33 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:INIADPUV1/OBJ=OBJ\$:INIADPUV1 MSRC\$:CPUSWUV1/UPDATE=(ENH\$:CPUSWUV1)+MSRC\$:INIADP/UPDATE=(ENH\$:INIADP)+EXECMLS/LIB

0396

**DIGITAL
CONFIDE**

**CORPORATION
PROPRIETARY**

0397

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY